

# Kapitel 3

## Rekursion und Induktion

### Abschnitt 3.1

#### Rekursive Definitionen

# Rekursive Definition von Funktionen über den natürlichen Zahlen

Eine Funktion  $f : \mathbb{N} \rightarrow Y$  (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.

## Beispiele

### Beispiel 3.1 (Fakultät)

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

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

Dann gilt  $f(n) = 1 \cdot 2 \cdot \dots \cdot n$  für alle  $n \in \mathbb{N}_+$

Man schreibt  $f(n) =: n!$ .

### Beispiel 3.2 (Fibonacci Folge)

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

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

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

## Beispiel 3.3 (Unäre Kodierung)

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

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

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

## Beispiel 3.4 (Binäre Kodierung)

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

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

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

# Rekursive Definition von Mengen

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

**Basisregeln** der Form

$$„x \in X“$$

**Rekursiven Regeln** der Form

$$„Wenn  $x_1, \dots, x_k \in X$ , dann  $x \in X$ “,$$

wobei  $x$  von  $x_1, \dots, x_k$  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  $\Sigma_{\text{ASCII}}$  das ASCII-Alphabet. Wir verwenden für Symbole aus  $\Sigma_{\text{ASCII}}$  einen Schreibmaschinenfont ( $A, B, \dots, a, b, \dots, 0, 1, \dots, (, ), \dots$ ).

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

Basisregeln.

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

Rekursive Regeln.

Für alle  $w \in \Sigma_{\text{ASCII}}^* \setminus \{0\}$ :

$$\text{Wenn } w \in \text{BIN}, \text{ dann } w0, w1 \in \text{BIN}.$$

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

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

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

Basisregel.

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

Rekursive Regeln.

Für alle  $v \in \text{BIN}$  und  $w \in \Sigma_{\text{ASCII}}^*$ :

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

Beispiele

$$\begin{aligned} \text{nil} &\in \text{LIST}(\text{BIN}), \\ \text{cons}(23, \text{cons}(0, \text{cons}(113, \text{nil}))) &\in \text{LIST}(\text{BIN}). \end{aligned}$$

## Beispiel 3.8: Klammersausdrücke

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

Basisregel.

$$\varepsilon \in L_K.$$

Rekursive Regeln.

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

Wenn  $w_1, \dots, w_k \in L_K$ , dann  $(w_1 w_2 \dots w_k) \in L_K$ .

Beispiele

$$\begin{aligned} \varepsilon, \quad (()), \quad (()(())) &\in L_K \\ ((), \quad (()), \quad ()) &\notin L_K \end{aligned}$$

Sei  $\Sigma$  ein Alphabet. Dann lässt sich  $\Sigma^*$  rekursiv wie folgt definieren.

Basisregel.

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

Rekursive Regeln.

Für alle  $w$  und alle  $a \in \Sigma$ :

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

## 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“?

### Definition 3.10

Eine **Ableitung** eines Elementes  $z$  in  $\mathcal{D}$  ist eine Folge  $(y_1, \dots, y_n)$ , 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  $x_1, \dots, x_k \in X$ , dann  $x \in X$ “ in  $\mathcal{D}$  und  $j_1, j_2, \dots, j_k < i$ , so dass  $y_{j_1} = x_1, y_{j_2} = x_2, \dots, y_{j_k} = x_k$  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):

$$(\varepsilon, (), (()), (()((()))).$$

Eine verständlichere Darstellung dieser Ableitung ist:

- |    |                |                         |
|----|----------------|-------------------------|
| 1. | $\varepsilon,$ | 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):

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

## Abgeschlossene Mengen

## 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  $x_1, \dots, x_k \in X$ , dann  $x \in X$ “ in  $\mathcal{D}$  gilt:

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

## Beispiel 3.14

Die Mengen  $\text{BIN}$ ,  $\Sigma_{\text{ASCII}}^*$  und

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

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

## 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*.

## Beweis von Satz 3.15 I

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  $x_1, \dots, x_k \in X$ , dann  $x \in X$ “ in  $\mathcal{D}$ , und nehmen wir an, dass  $x_1, \dots, x_k \in X$ . Für  $1 \leq i \leq k$  sei

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

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

Dann ist

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

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



## Beweis von Satz 3.15 II

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

Sei also  $z \in X$ , und sei

$$y_1, \dots, y_n$$

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:**  $i \rightarrow i + 1$ , wobei  $i + 1 \leq n$ .

Wenn  $\mathcal{D}$  eine Basisregel „ $x \in X$ “ mit  $x = y_{i+1}$  enthält, so gilt wie im Induktionsanfang  $y_{i+1} = x \in Y$ .

## Beweis von Satz 3.15 III

Sonst enthält  $\mathcal{D}$  eine rekursive Regel „Wenn  $x_1, \dots, x_k \in X$ , dann  $x \in X$ “, und es gibt  $j_1, j_2, \dots, j_k \leq i$ , so dass  $y_{j_1} = x_1, y_{j_2} = x_2, \dots, y_{j_k} = x_k$  und  $y_{i+1} = x$ .

Nach Induktionsannahme sind  $y_{j_1}, y_{j_2}, \dots, y_{j_k} \in Y$ .

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.

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

**Rekursionsanfang.**

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

**Rekursionsschritt.**

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

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.

## Beispiele

### Beispiel 3.16 (Länge eines Wortes)

Sei  $\Sigma$  ein Alphabet. Wir definieren die Funktion  $|\cdot| : \Sigma^* \rightarrow \mathbb{N}$  rekursiv über die rekursive Definition von  $\Sigma^*$  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  $\text{lg} : \text{LIST}(\text{BIN}) \rightarrow \mathbb{N}$  rekursiv über den Aufbau von  $\text{LIST}(\text{BIN})$ :

**Rekursionsanfang.**  $\text{lg}(\text{nil}) := 0$ .

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

$$\text{lg}(\text{cons}(v, w)) := \text{lg}(w) + 1.$$

## Beispiel 3.18 (Wert einer Binärzahl)

Wir definieren die Funktion  $\text{val}_{\text{BIN}} : \text{BIN} \rightarrow \mathbb{N}$  rekursiv über den Aufbau von BIN:

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

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

$$\begin{aligned}\text{val}_{\text{BIN}}(v0) &:= 2 \text{val}_{\text{BIN}}(v), \\ \text{val}_{\text{BIN}}(v1) &:= 2 \text{val}_{\text{BIN}}(v) + 1.\end{aligned}$$

## Beispiel 3.19 (Summation über eine Liste)

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

**Rekursionsanfang.**  $\text{sum}(\text{nil}) := 0.$

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

$$\text{sum}(\text{cons}(v, w)) := \text{val}_{\text{BIN}}(v) + \text{sum}(w).$$

## Beispiel 3.20: Arithmetische Terme

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

**Basisregeln.**

Für alle  $v \in \text{BIN}$ :

$$v \in \text{ART}$$

**Rekursive Regeln.**

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

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

## Beispiele

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

Wir definieren die Funktion  $\text{val}_{\text{ART}} : \text{ART} \rightarrow \mathbb{Z}$  rekursiv über den Aufbau von ART:

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

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

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

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

## Beispiel 3.20 (Forts.)

Es gilt

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

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

Daraus ergibt sich

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

## 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.

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  $x_1, \dots, x_k \in X$ , dann  $x \in X$ “, beweise  $A(x)$  unter Verwendung der **Induktionsannahmen**  $A(x_1), \dots, A(x_k)$ .

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  $X$** .

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  $\mathcal{D}$ .

## Beispiel 3.21 |

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  $w_1, \dots, w_k \in \Sigma^*$ :

Wenn  $w_1, \dots, w_k \in L_K$ , dann  $(w_1 w_2 \dots w_k) \in L_K$ .

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

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

Alternativ können wir die Funktionen rekursiv definieren durch

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

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

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

### Beweis

# Beispiel 3.21 III

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

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

**Induktionsanfang:**  $w = \varepsilon$

Es gilt  $\text{auf}(\varepsilon) = \text{zu}(\varepsilon) = 0$ , und weil  $\varepsilon$  keine echten Präfixe hat, auch  $\text{auf}(v) > \text{zu}(v)$  für alle  $v \sqsubset w$ .

**Induktionsschritt:**  $w = (w_1 \dots w_k)$  für ein  $k \geq 1$  und  $w_1, \dots, w_k \in L_K$ .

Nach Induktionsannahme gilt  $(\star)$  für alle  $w_i$ .

Damit

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

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

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

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

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

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

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

## Beispiel 3.21 V

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

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

**Induktionsanfang:**  $n = 0$ .

Dann ist  $w = \varepsilon \in L_K$ .

**Induktionsschritt:**  $n > 0$ .

Sei  $w = a_1 \dots a_n \in \Sigma^*$ , so dass

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

**Induktionsannahme:** (★★) 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  $\text{auf}(a_1) \geq \text{zu}(a_1)$ .



- ▶ Es gilt  $a_n = )$ , weil  $\text{auf}(a_1 \dots a_{n-1}) > \text{zu}(a_1 \dots a_{n-1})$  und  $\text{auf}(a_1 \dots a_n) = \text{zu}(a_1 \dots a_n)$ .
- ▶ Seien  $1 \leq i_0 < i_1 < \dots < i_k \leq n$  so, dass für  $0 \leq j \leq k$

$$\text{auf}(a_1 \dots a_{i_j}) = \text{zu}(a_1 \dots a_{i_j}) + 1$$

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

$$\text{auf}(a_1 \dots a_i) > \text{zu}(a_1 \dots a_i) + 1.$$

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

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

Dann gilt

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

## Beispiel 3.21 VII

Für  $1 \leq j \leq k$  gilt

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

Sei nun  $v \sqsubset w_j$  mit  $v \neq \varepsilon$ . Dann existiert ein

$i \in \{i_{j-1} + 1, \dots, i_j - 1\}$ , so dass  $v = a_{i_{j-1}+1} \dots a_i$ . Es gilt

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

Nach Induktionsannahme gilt dann  $w_j \in L_K$ .

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



## Beispiel 3.23 I

Wir definieren rekursiv eine **Nachfolgerfunktion**  $N : \text{BIN} \rightarrow \Sigma_{\text{ASCII}}^*$ :

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

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

$$\begin{aligned} N(v0) &:= v1, \\ N(v1) &:= N(v)0. \end{aligned}$$

### Behauptung 1

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

**Beweis.** Induktion über den Aufbau von BIN.

**Induktionsanfang.**

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

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

Induktionsschritt.

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

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

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

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

Außerdem gilt

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



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

## Rekursionsanfang.

- Definition  $A(0, v)$  für alle  $v \in \text{BIN}$ :

$$A(0, v) := v.$$

- Definition  $A(1, v)$  für alle  $v \in \text{BIN}$ :

$$A(1, v) := N(v).$$

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

# Beispiel 3.23 V

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

## Rekursionsanfang.

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

**Rekursionsschritt.** Für alle  $v' \in \text{BIN} \setminus \{0\}$ :

$$A(u'0, v'0) := A(u', v')0,$$

$$A(u'0, v'1) := A(u', v')1.$$

- Rekursive Definition  $A(u'1, v)$  für alle  $v \in \text{BIN}$ .

Rekursionsanfang.

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

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

$$\begin{aligned} A(u'1, v'0) &:= A(u', v')1, \\ A(u'1, v'1) &:= N(A(u', v'))0. \end{aligned}$$

## Behauptung 2

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

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

Wir verzichten auf einen Beweis.