

Einführung in die Programmierung  
WS 2018/19

Übungsblatt 2: Boolesche Algebra, Vollständige Induktion

Besprechung: 05.11.2018 - 09.11.2018

**Aufgabe 2-1**      *Boolsche Operatoren*

Aus der Vorlesung kennen Sie die boolschen Operatoren  $\wedge : \mathbb{B}^2 \rightarrow \mathbb{B}$ ,  $\vee : \mathbb{B}^2 \rightarrow \mathbb{B}$  sowie  $\neg : \mathbb{B} \rightarrow \mathbb{B}$ . Mittels funktionaler Komposition lassen sich weitere Operatoren definieren, zum Beispiel gilt:

$$\implies : \mathbb{B}^2 \rightarrow \mathbb{B}, (x, y) \mapsto \neg x \vee y$$

Außerdem lassen sich dadurch Operatoren höherer Stelligkeit definieren, z.B. Negation der ersten und zweiten Komponente:

$$\neg_1 : \mathbb{B}^2 \rightarrow \mathbb{B}, (x, y) \mapsto \neg x$$

$$\neg_2 : \mathbb{B}^2 \rightarrow \mathbb{B}, (x, y) \mapsto \neg y$$

Sei nun  $\mathfrak{B}_2 = \{f \in \mathbb{B}^2 \rightarrow \mathbb{B}\}$  die Menge aller 2-stelligen totalen boolschen Funktionen.

- (a) Benutzen Sie die Junktorenmenge  $\{\neg_1, \wedge\} \subset \mathfrak{B}_2$  und konstruieren Sie daraus  $\vee \in \mathfrak{B}_2$  durch Komposition.

*Hinweis: DeMorgansche Regeln helfen.*

- (b)  $\uparrow$  (NAND) ist wie folgt definiert:

$$x \uparrow y = \neg(x \wedge y)$$

Zeigen Sie, dass sich aus  $\{\uparrow\}$  die Junktorenmenge  $\{\neg_1, \wedge, \vee\}$  konstruieren lässt.

*Hinweis: Zeigen Sie die Herleitungen in der angegebenen Reihenfolge  $\neg_1, \wedge, \vee$  und nutzen Sie Ergebnisse aus der vorherigen Aufgabe.*

- (c) Geben Sie die Mächtigkeit von  $\mathfrak{B}_2$  an. Wieviele 3-stellige boolsche Funktionen  $|\mathfrak{B}_3|$  gibt es?

*Hinweis: Wieviele Wahrheitstabellen kann es für zwei Inputs  $x, y$  geben?*

**Aufgabe 2-2**      *Vollständige Induktion*

- (a) Zeigen Sie mittels vollst. Induktion:  $1 + 2 + 4 + 8 + \dots + 2^n = 2^{n+1} - 1$

- (b) Sei  $a$  eine Folge, die folgendermaßen rekursiv definiert ist:

$$a_1 = 2$$

$$a_{n+1} = 2 - \frac{1}{a_n}$$

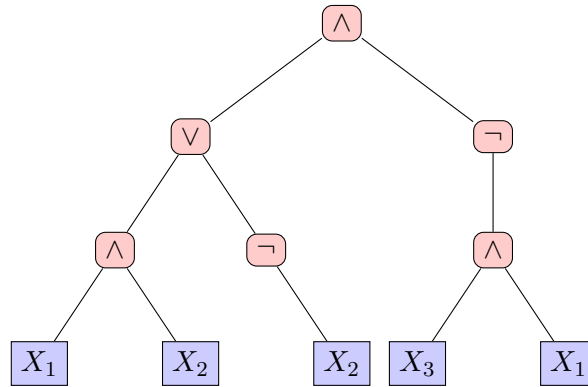
Zeigen Sie mittels vollst. Induktion:  $a_n = \frac{n+1}{n}$ .

### Aufgabe 2-3 *Bäume*

Sei  $B = \{\neg, \wedge, \vee\} \subset \mathfrak{B}$ . Außerdem sei  $\Sigma = \{X_1, \dots, X_n\}$  eine endliche Menge von logischen Literalen. Diese können einen beliebigen Wahrheitswert aus  $\{\text{True}, \text{False}\}$  annehmen. Wir definieren darüber die Menge der Wahrheitsbäume  $T_B$  induktiv:

$$\begin{aligned}\Sigma &\subseteq T_B \\ \neg(t) &\in T_B \text{ für } t \in T_B \\ \wedge(t_1, t_2) &\in T_B \text{ für } t_1, t_2 \in T_B : \\ \vee(t_1, t_2) &\in T_B \text{ für } t_1, t_2 \in T_B :\end{aligned}$$

Damit sind Blattknoten stets Literale und innere Knoten logische Junktoren. Als Beispiel soll folgende Darstellung dienen:



Der gezeigte Baum kann sequentiell dargestellt werden als

$$t' = \wedge(\vee(\wedge(X_1, X_2), \neg(X_2)), \neg(\wedge(X_3, X_1)))$$

Wir können nun eine rekursive Funktion definieren, die die induktive Konstruktion eines Baumes nutzt, um seinen Wahrheitswert zu bestimmen:

$$\begin{aligned}\text{evaluate} : T_B &\rightarrow \mathbb{B} \\ \text{evaluate}(X \in \Sigma) &= X \\ \text{evaluate}(\neg(t)) &= \neg \text{evaluate}(t) \\ \text{evaluate}(\wedge(t_1, t_2)) &= \text{evaluate}(t_1) \wedge \text{evaluate}(t_2) \\ \text{evaluate}(\vee(t_1, t_2)) &= \text{evaluate}(t_1) \vee \text{evaluate}(t_2)\end{aligned}$$

- (a) Ist die Funktion  $\text{evaluate}(t)$  injektiv/surjektiv/bijektiv?
- (b) Definieren Sie eine rekursive Funktion, die die Höhe eines Baumes zurückgibt:

$$\text{height} : T_B \rightarrow \mathbb{N}$$

Zur Vereinheitlichung:  $\text{height}(X \in \Sigma) = 0$

- (c) Eine Formel ist in Negationsnormalform (NNF), wenn Negationen nur noch direkt vor den atomaren Aussagen auftauchen. In unserem Fall bedeutet dies, dass Negationen nur unmittelbar vor den Blattknoten auftreten dürfen. Der oben gezeigte Baum  $t'$  erfüllt dies nicht. Es gilt aber

$$\text{NNF}(t') = \wedge(\vee(\wedge(X_1, X_2), \neg(X_2)), \vee(\neg(X_3), \neg(X_1))).$$

Definieren Sie nun eine rekursive Funktion  $\text{NNF} : T_B \rightarrow T_B$ , die einen Baum so transformiert, dass das Resultat in NNF vorliegt.