



Lehrstuhl für  
**Eingebettete Systeme**



**Entwurfsmethodik**  
Institut für Informatik

# Hardwarearchitekturen und Rechensysteme

## 3. Boolesche Funktionen und Boolesche Algebra

Folien zur Vorlesung Hardwarearchitekturen und Rechensysteme von

Prof. Dr. rer. Nat. U. Brinkschulte

**Prof. Dr.-Ing. L. Hedrich**

Prof. Dr.-Ing. K. Waldschmidt

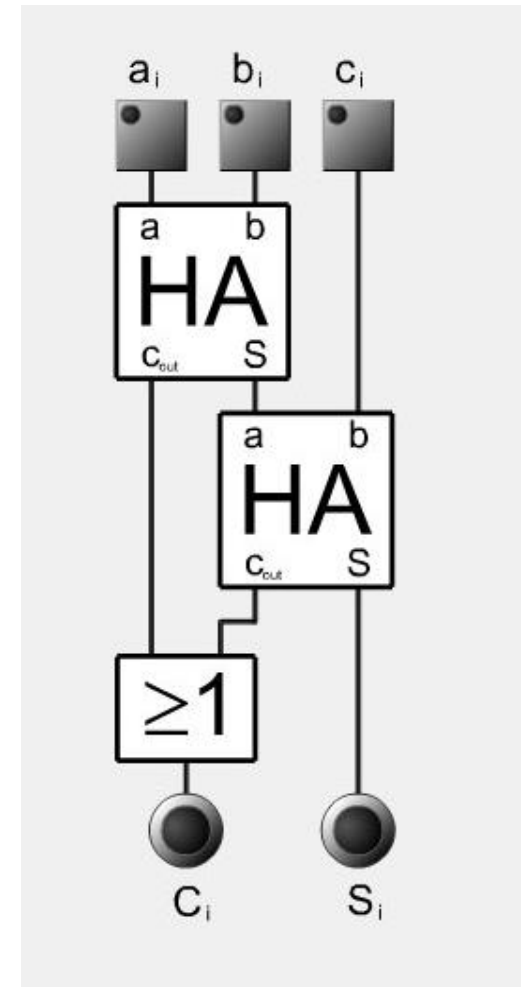
# Motivation

Die Boolesche Algebra wird benötigt für die Modellierung digitaler Schaltnetze und Schaltwerke, insbesondere für die

- Beschreibung,
- Analyse und
- Optimierung

digitaler Schaltungen.

Das Kapitel „Aussagenlogik“ aus der Vorlesung diskrete Modellierung ergänzt die Inhalte dieses Kapitels.



# Gliederung

## 3.1 Boolesche Funktionen

## 3.2 Boolesche Algebra

## 3.3 Darstellung Boolescher Funktionen

### 3.3.1 KV-Diagramme

### 3.3.2 Schaltpläne

### 3.3.3 Zweistufige Normalformen

### 3.3.4 Vektordarstellung

### 3.3.5 Binäre Entscheidungsgraphen

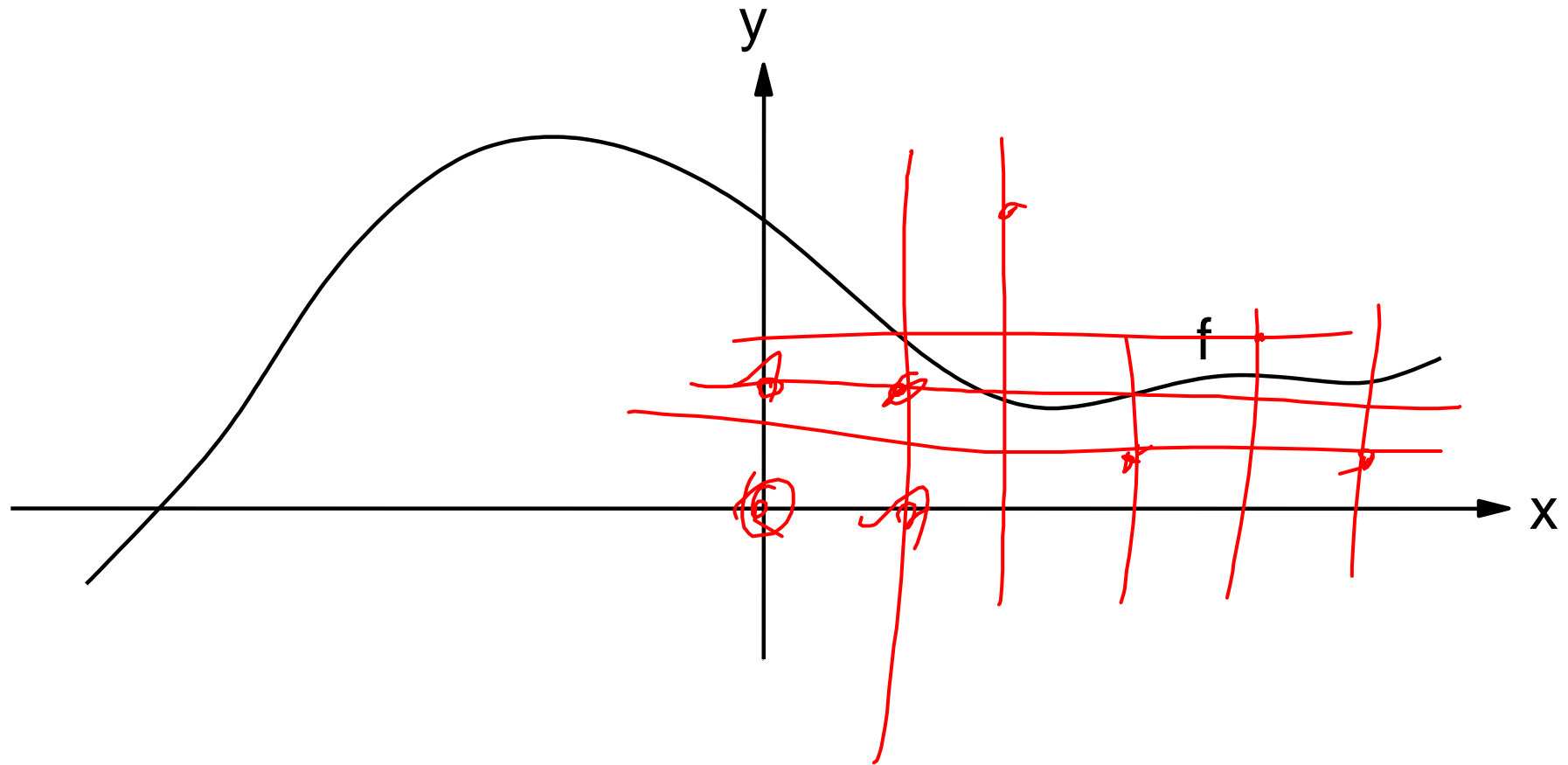
## 3.4 Optimierung Boolescher Funktionen

### 3.4.1 Graphisches Verfahren

### 3.4.2 Verfahren von Quine/McCluskey

# 3.1 Boolesche Funktionen

Reelle Funktionen einer Variablen:



# Reelle und Boolesche Funktionen

## Reelle Funktionen in $n$ Variablen:

$$y = f(x_0, \dots, x_{n-1}) \quad y, x_i \in \mathbb{R} \quad \text{oder} \quad f : \mathbb{R}^n \rightarrow \mathbb{R}$$

$$\mathbb{R}^n := \underbrace{\mathbb{R} \times \mathbb{R} \times \dots \times \mathbb{R}}_n \quad (\text{n-faches Kreuzprodukt})$$

$$A = \{a_0, \dots, a_{k-1}\} \quad B = \{b_0, \dots, b_{l-1}\}$$

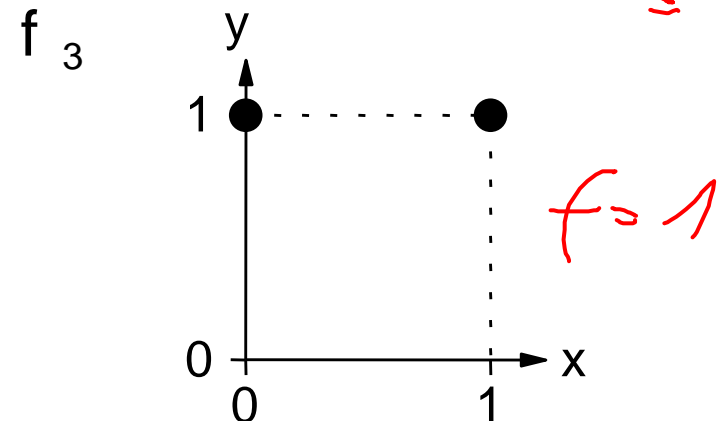
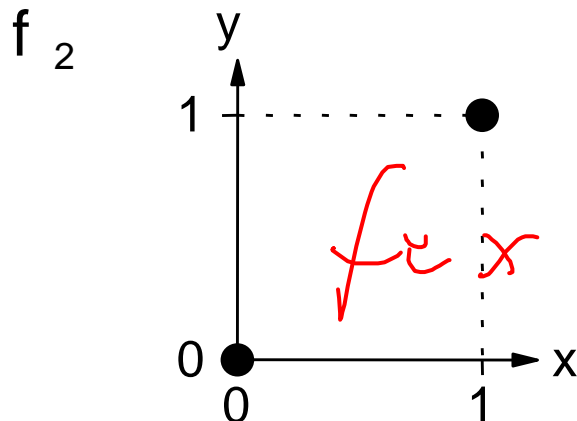
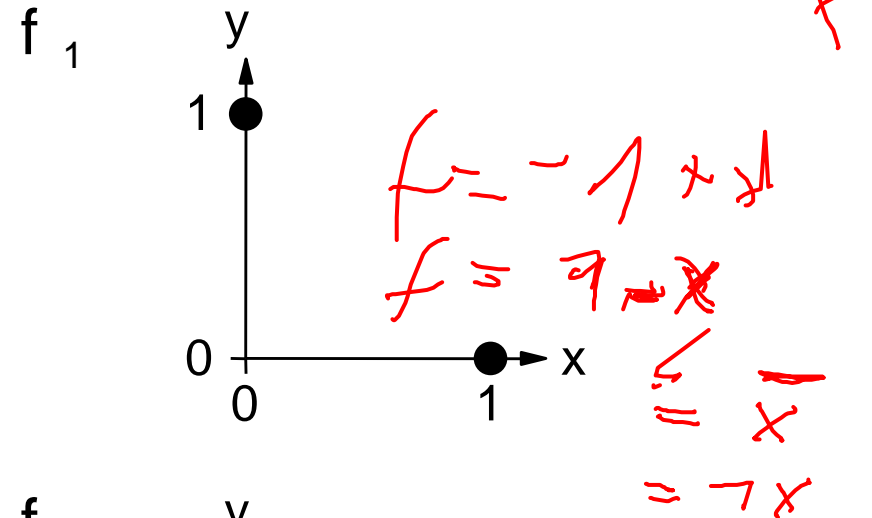
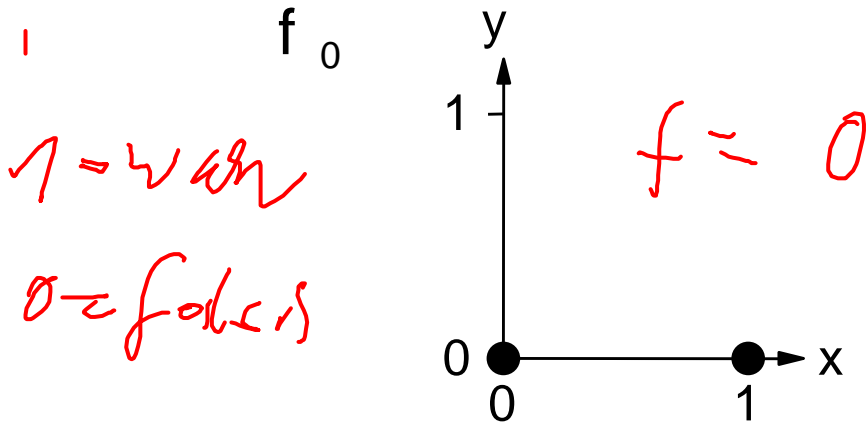
$$A \times B := \underbrace{\{(a_0, b_0), \dots, (a_0, b_{l-1}), \dots, (a_{k-1}, b_0), \dots, (a_{k-1}, b_{l-1})\}}_{k \cdot l}$$

## Boolesche Funktionen in $n$ Variablen:

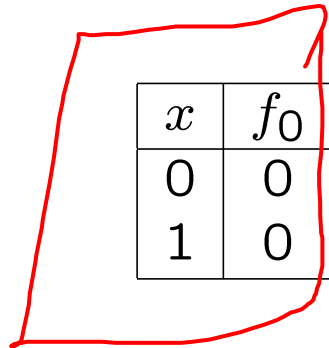
$$y = f(x_0, \dots, x_{n-1}) \quad y, x_i \in \mathbb{B} = \{0, 1\} \quad \text{oder} \quad f : \mathbb{B}^n \rightarrow \mathbb{B}$$

# Funktion einer Variablen

Alle Booleschen Funktionen einer Variablen:



# Alle Booleschen Funktionen einer Variablen



$x$	$f_0$	$f_1$	$f_2$	$f_3$
0	0	1	0	1
1	0	0	1	1

$$f_0(x) = 0$$

$$f_1(x) = \bar{x}$$

$$f_2(x) = x$$

$$f_3(x) = 1$$

$\bar{x}$  heißt Negation von  $x$ . Bei analoger Interpretation des Minuszeichens wäre die Schreibweise  $\bar{x} = 1 - x$  ebenfalls erlaubt.

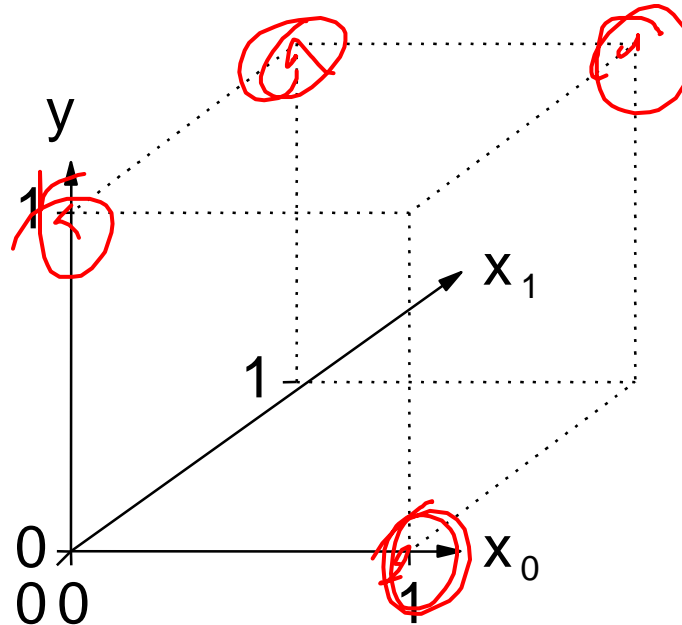
Für Funktionen gilt entsprechend:

$$\bar{f}(x_0, x_1, \dots, x_{n-1}) = 1 - f(x_0, x_1, \dots, x_{n-1})$$

## Literal:

Die Größe  $\underline{x} \in \{x, \bar{x}\}$  heißt Literal. Literale sind als Quasivariablen zu verstehen.

# Einheitswürfel



Die Funktionswerte  $y$  aller Funktionen  $f_i(x_0; x_1)$  können nur die Eckpunkte des Einheitswürfels belegen.

- Für jede Belegung der Funktionsargumente kann eine Boolesche Funktion genau einen von zwei Werten annehmen.
- Es existieren insgesamt  $2^4 = 16$  verschiedene Boolesche Funktionen mit zwei Variablen.



# Booleschen Funktionen von zwei Variablen

Alle Booleschen Funktionen von zwei Variablen:

$x_0$	$x_1$	$f_0$	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
0	0	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0

$x_0$	$x_1$	$f_8$	$f_9$	$f_{10}$	$f_{11}$	$f_{12}$	$f_{13}$	$f_{14}$	$f_{15}$
0	0	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1

$\oplus$   
NAND  
 $\overline{x_0 \wedge x_1}$

A'gavilz ler =

# Boolesche Funktionen von zwei Variablen

## Prominente Funktionen zweier Variablen:

Funktion	Bedeutung	Infix-Schreibweise
$f_8(x_0, x_1)$	Konjunktion (AND)	$x_0 \wedge x_1$
$f_{14}(x_0, x_1)$	Disjunktion (OR)	$x_0 \vee x_1$
$f_7(x_0, x_1)$	NAND	$\overline{x_0 \wedge x_1}$
$f_1(x_0, x_1)$	NOR	$\overline{x_0 \vee x_1}$
$f_{11}(x_0, x_1)$	Implikation	$x_0 \Rightarrow x_1$
$f_9(x_0, x_1)$	Äquivalenz	$x_0 \Leftrightarrow x_1$
$f_6(x_0, x_1)$	Antivalenz (XOR)	$x_0 \oplus x_1$

• d □

+ +

$\overline{x}$   $\Rightarrow$   $\neg x$

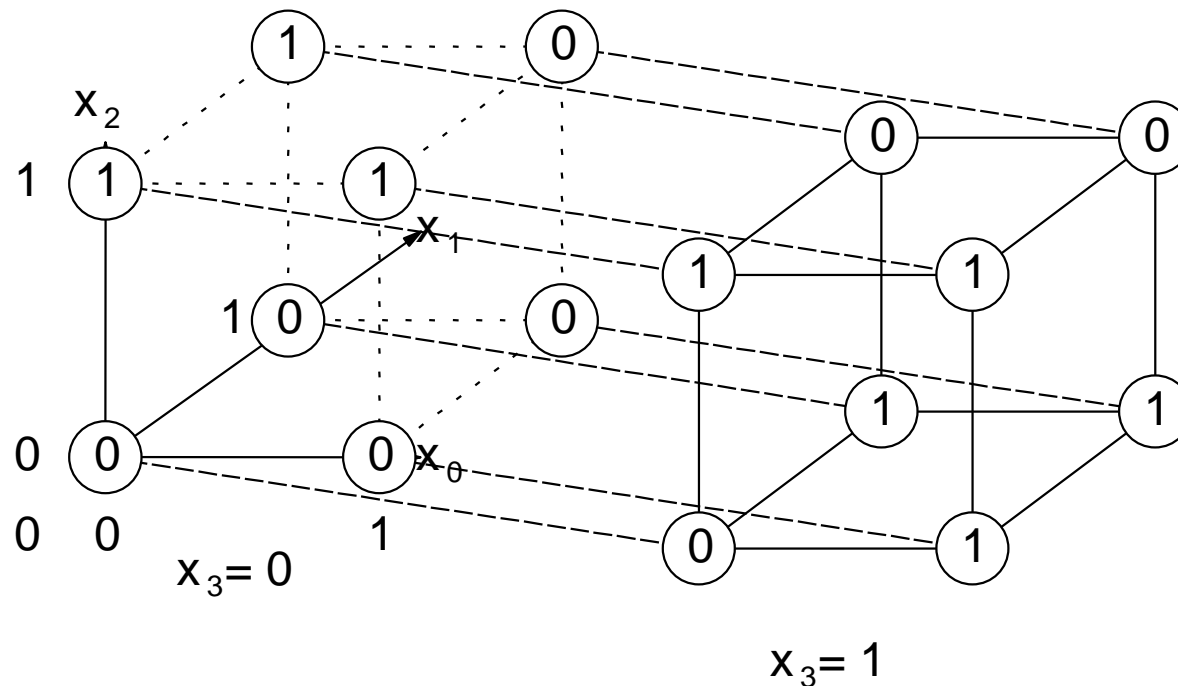
In der Praxis kommen NAND- und NOR-Gatter häufig vor, weil sie technologisch einfach zu realisieren sind.

# Bemerkungen zu Booleschen Funktionen

- Die in der Infix-Schreibweise verwendeten Booleschen Operatoren entstammen der Booleschen Algebra.
- Die Negation besitzt die höchste Priorität.
- Die Konjunktion ( $\wedge$ ) besitzt eine höhere Priorität als die Disjunktion ( $\vee$ ) .  
$$x_0 \vee x_1 \wedge x_2 = x_0 \vee (x_1 \wedge x_2) \neq (x_0 \vee x_1) \wedge x_2$$
- Oft wird in Booleschen Ausdrücken der Konjunktionsoperator ( $\wedge$ ) weggelassen.  
$$x_0 \wedge x_1 = x_0 x_1$$
- Die Äquivalenz entspricht der Negation der Antivalenz.  
$$x_0 \Leftrightarrow x_1 = \overline{x_0 \oplus x_1}$$
 ↔ ~~(x ⊕ x)~~ (x ∧ y) ∧ c
- Die Antivalenz entspricht der einstelligen dualen Addition.
- Die Konjunktion entspricht dem Übertrag der einstelligen Addition.
- Gleichheit ( $x_0 = x_1$ ) und Äquivalenz ( $x_0 \Leftrightarrow x_1$ ) sollten nicht verwechselt werden.

# Funktionen von $n$ Variablen

Auch bei mehr als zwei Variablen ist eine graphische Darstellung aufgrund des diskreten Wertebereichs möglich. Diese Darstellungen werden jedoch schnell unübersichtlich.



Vierdimensionaler Hyperwürfel mit eingetragenen Funktionswerten für genau eine Funktion

# Funktionen von $n$ Variablen

- Zur Darstellung Boolescher Funktionen eignen sich Funktionstabeln, in denen für jede Eingangsbelegung  $x$  der zugehörige Funktionswert notiert wird.
- Die Funktionstafel einer Booleschen Funktion in  $n$  Variablen hat genau  $2^n$  Zeilen.

## Funktionstafel:

Nr.	$x$ ( $n$ -stelliger Vektor)	$f$
0	0 ... 00	$W_{f,0}$
1	0 ... 01	$W_{f,1}$
2	0 ... 10	$W_{f,2}$
⋮	⋮	⋮
$2^n - 1$	1 ... 11	$W_{f,2^n-1}$

$W_{f,i} \in \{0, 1\}$  sind die Wahrheitswerte der Funktion.

# Bemerkungen

- Es existieren  $2^{2^n}$  verschiedene Funktionen von  $n$  Variablen. Eine Funktionsspalte hat  $2^n$  Zeilen. Die Werte können als Dualzahl der Länge  $2^n$  interpretiert werden. Eine Dualzahl der Länge  $l$  kann aber  $2^l$  verschiedene Werte annehmen.
- Sind zwei Funktionen  $f_1$  und  $f_2$  voneinander verschieden, so sind auch deren Inverse voneinander verschieden.

$$W_{f_1,i} \neq W_{f_2,i} \quad \Rightarrow \quad \overline{W_{f_1,i}} \neq \overline{W_{f_2,i}}$$

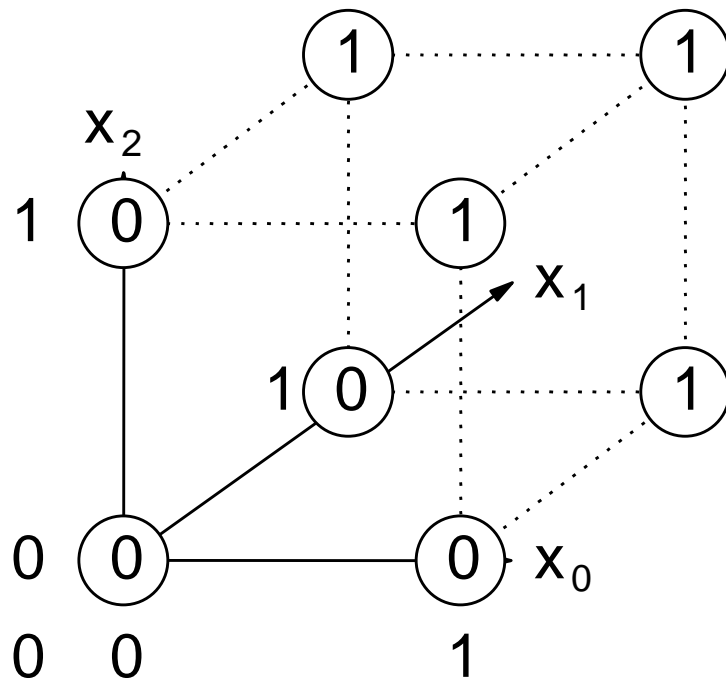
# Beispiel - Die 2-von-3 Funktion

Die 2-von-3 Funktion liefert genau dann eine 1, wenn mindestens 2 der 3 Eingangsvariablen den Wert 1 haben.

**Funktionstafel:**

$x_2$	$x_1$	$x_0$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

**3-dimensionaler Hyperwürfel:**



## 3.2 Boolesche Algebra

Eine Verknüpfungsstruktur  $\mathbb{B} = \{T; \wedge; \vee; \neg\}$ , auf deren Trägermenge  $T$  zwei zweistellige Verknüpfungen definiert sind und eine einstellige Verknüpfung (Negation), heißt genau dann *Boolesche Algebra*, wenn die folgenden Axiome erfüllt sind.

- |   |  |   |
|---|--|---|
| <b>0. Abgeschlossenheit :</b>                 | $\wedge$ und $\vee$ sind abgeschlossen auf $T$         |   |
| <b>1. Kommutativgesetz der Konjunktion :</b>  | $a \wedge b = b \wedge a$                              | • |
| <b>2. Kommutativgesetz der Disjunktion :</b>  | $a \vee b = b \vee a$                                  | ✗ |
| <b>3. Assoziativgesetz der Konjunktion :</b>  | $(a \wedge b) \wedge c = a \wedge (b \wedge c)$        |   |
| <b>4. Assoziativgesetz der Disjunktion :</b>  | $(a \vee b) \vee c = a \vee (b \vee c)$                |   |
| <b>5. Erstes Distributivgesetz :</b>          | $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ |   |
| <b>6. Zweites Distributivgesetz :</b>         | $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$   |   |
| <b>7. Identitätselement der Konjunktion :</b> | $1 \wedge a = a$                                       |   |
| <b>8. Identitätselement der Disjunktion :</b> | $0 \vee a = a$   |   |
| <b>9. Inverses Element der Konjunktion :</b>  | $a \wedge \bar{a} = 0$                                 |   |
| <b>10. Inverses Element der Disjunktion :</b> | $a \vee \bar{a} = 1$                                   |   |



# Idempotenzgesetze

**11. Idempotenzgesetz der Konjunktion:**  $a \wedge a = a$

**12. Idempotenzgesetz der Disjunktion:**  $a \vee a = a$

**Beweis:**

11. Idempotenzgesetz der Konjunktion

$$\begin{aligned} a &\stackrel{7}{=} 1 \wedge a && \stackrel{1}{=} a \wedge 1 && \stackrel{10}{=} a \wedge (a \vee \bar{a}) \\ &\stackrel{5}{=} (a \wedge a) \vee (a \wedge \bar{a}) && \stackrel{9}{=} (a \wedge a) \vee 0 && \stackrel{2}{=} 0 \vee (a \wedge a) && \stackrel{8}{=} a \wedge a \end{aligned}$$

12. Idempotenzgesetz der Disjunktion

$$\begin{aligned} a &\stackrel{8}{=} 0 \vee a && \stackrel{2}{=} a \vee 0 && \stackrel{9}{=} a \vee (a \wedge \bar{a}) \\ &\stackrel{6}{=} (a \vee a) \wedge (a \vee \bar{a}) && \stackrel{10}{=} (a \vee a) \wedge 1 && \stackrel{1}{=} 1 \wedge (a \vee a) && \stackrel{7}{=} a \vee a \end{aligned}$$

# Reduktionsgesetze

13. Reduktionsgesetz der Konjunktion  $a \wedge 0 = 0$

14. Reduktionsgesetz der Disjunktion  $a \vee 1 = 1$

**Beweis:**

13. Reduktionsgesetz der Konjunktion

$$0 \stackrel{9}{=} a \wedge \bar{a} \quad \stackrel{8}{=} a \wedge (0 \vee \bar{a}) \stackrel{5}{=} (a \wedge 0) \vee (a \wedge \bar{a})$$

$$\stackrel{9}{=} (a \wedge 0) \vee 0 \stackrel{2}{=} 0 \vee (a \wedge 0) \stackrel{8}{=} a \wedge 0$$

14. Reduktionsgesetz der Disjunktion

$$1 \stackrel{10}{=} a \vee \bar{a} \quad \stackrel{7}{=} a \vee (1 \wedge \bar{a}) \stackrel{6}{=} (a \vee 1) \wedge (a \vee \bar{a})$$

$$\stackrel{10}{=} (a \vee 1) \wedge 1 \stackrel{1}{=} 1 \wedge (a \vee 1) \stackrel{7}{=} a \vee 1$$

# Eindeutigkeit des inversen Elements

Zu jedem  $a \in T$  existiert *genau ein* inverses Element  $\bar{a} \in T$ , so dass gilt:

$$a \wedge \bar{a} = 0 \text{ und } a \vee \bar{a} = 1$$

## Beweis:

Annahme:  $\bar{a}_1$  und  $\bar{a}_2$  seien invers zu  $a$ .

$$\begin{aligned} \bar{a}_1 &\stackrel{7}{=} 1 \wedge \bar{a}_1 && \stackrel{1}{=} \bar{a}_1 \wedge 1 && \stackrel{10}{=} \bar{a}_1 \wedge (a \vee \bar{a}_2) \\ &\stackrel{5}{=} (\bar{a}_1 \wedge a) \vee (\bar{a}_1 \wedge \bar{a}_2) && \stackrel{1}{=} (a \wedge \bar{a}_1) \vee (\bar{a}_1 \wedge \bar{a}_2) && \stackrel{9}{=} 0 \vee (\bar{a}_1 \wedge \bar{a}_2) \\ &\stackrel{8}{=} \bar{a}_1 \wedge \bar{a}_2 && \stackrel{1}{=} \bar{a}_2 \wedge \bar{a}_1 && \stackrel{8}{=} 0 \vee (\bar{a}_2 \wedge \bar{a}_1) \\ &\stackrel{9}{=} (a \wedge \bar{a}_2) \vee (\bar{a}_2 \wedge \bar{a}_1) && \stackrel{1}{=} (\bar{a}_2 \wedge a) \vee (\bar{a}_2 \wedge \bar{a}_1) && \stackrel{5}{=} \bar{a}_2 \wedge (a \vee \bar{a}_1) \\ &\stackrel{10}{=} \bar{a}_2 \wedge 1 && \stackrel{1}{=} 1 \wedge \bar{a}_2 && \stackrel{7}{=} \bar{a}_2 \end{aligned}$$

Aus der Eindeutigkeit des inversen Elements folgt:  $\bar{\bar{a}} = a$

# Absorptionsregeln

$$a) \quad a \vee (a \wedge b) = a$$

$$b) \quad a \wedge (a \vee b) = a$$

$$c) \quad (a \wedge b) \vee (a \wedge \bar{b}) = a$$

$$d) \quad (a \vee \bar{b}) \wedge b = a \wedge b$$

$$e) \quad (a \wedge \bar{b}) \vee b = a \vee b$$

$$f) \quad (a \vee b) \wedge (a \vee \bar{b}) = a$$

Beweis:

$$\begin{aligned} a) \quad a &\stackrel{7}{=} 1 \wedge a && \stackrel{14}{=} (b \vee 1) \wedge a && \stackrel{1}{=} a \wedge (b \vee 1) \\ &\stackrel{5}{=} (a \wedge b) \vee (a \wedge 1) && \stackrel{1}{=} (a \wedge b) \vee (1 \wedge a) && \stackrel{7}{=} (a \wedge b) \vee a && \stackrel{2}{=} a \vee (a \wedge b) \end{aligned}$$

$$\begin{aligned} b) \quad a &\stackrel{8}{=} 0 \vee a && \stackrel{13}{=} (b \wedge 0) \vee a && \stackrel{2}{=} a \vee (b \wedge 0) \\ &\stackrel{6}{=} (a \vee b) \wedge (a \vee 0) && \stackrel{2}{=} (a \vee b) \wedge (0 \vee a) && \stackrel{8}{=} (a \vee b) \wedge a && \stackrel{1}{=} a \wedge (a \vee b) \end{aligned}$$

# Zusammenfassung

- |                                      |  |
|--------------------------------------|--|
| 1. Kommutativgesetz der Konjunktion  | $a \wedge b = b \wedge a$                              |
| 2. Kommutativgesetz der Disjunktion  | $a \vee b = b \vee a$                                  |
| 3. Assoziativgesetz der Konjunktion  | $(a \wedge b) \wedge c = a \wedge (b \wedge c)$        |
| 4. Assoziativgesetz der Disjunktion  | $(a \vee b) \vee c = a \vee (b \vee c)$                |
| 5. Erstes Distributivgesetz          | $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ |
| 6. Zweites Distributivgesetz         | $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$   |
| 7. Identitätselement der Konjunktion | $1 \wedge a = a$                                       |
| 8. Identitätselement der Disjunktion | $0 \vee a = a$   |
| 9. Inverses Element der Konjunktion  | $a \wedge \bar{a} = 0$                                 |
| 10. Inverses Element der Disjunktion | $a \vee \bar{a} = 1$                                   |
| 11. Idempotenzgesetz der Konjunktion | $a \wedge a = a$                                       |
| 12. Idempotenzgesetz der Disjunktion | $a \vee a = a$   |
| 13. Reduktionsgesetz der Konjunktion | $a \wedge 0 = 0$                                       |
| 14. Reduktionsgesetz der Disjunktion | $a \vee 1 = 1$   |

# Gesetze von DeMorgan

Gesetze:

$$\overline{a \vee b} = \bar{a} \wedge \bar{b}$$

$$\overline{a \wedge b} = \bar{a} \vee \bar{b}$$

Wahrheitstafel:

$a$	$b$	$\bar{a}$	$\bar{b}$	$\overline{a \vee b}$	$\bar{a} \wedge \bar{b}$	$\overline{a \wedge b}$	$\bar{a} \vee \bar{b}$
0	0	1	1	1	1	1	1
0	1	1	0	0	0	1	1
1	0	0	1	0	0	1	1
1	1	0	0	0	0	0	0

Diese beiden Regeln haben große Bedeutung für die Umformung von Booleschen Ausdrücken. Sie lassen sich auf beliebige Variablenzahl verallgemeinern:

$$\overline{\bigvee_{i=0}^{n-1} x_i} = \bigwedge_{i=0}^{n-1} \bar{x}_i$$

und

$$\overline{\bigwedge_{i=0}^{n-1} x_i} = \bigvee_{i=0}^{n-1} \bar{x}_i$$

# Der Shannon'sche Inversionssatz

Jede nur mit den Operatoren für AND, OR und NOT gebildete Schaltfunktion kann dadurch negiert werden, dass die Operatoren für AND und OR miteinander vertauscht werden und jedes Literal negiert wird. Außerdem sind 0 und 1 zu vertauschen.

## Beispiel:

$$\begin{aligned} f(a, b, c, d) &= ac\bar{d} \vee (\bar{a}b \vee c)(\bar{b}d \vee \bar{c}) \\ &= (ac\bar{d}) \vee (((\bar{a}b) \vee c)((\bar{b}d) \vee \bar{c})) \end{aligned}$$

$$\Rightarrow \bar{f}(a, b, c, d) = (\bar{a} \vee \bar{c} \vee d)((a \vee \bar{b})\bar{c}) \vee ((b \vee \bar{d})c)$$

## Hinweis:

In Booleschen Ausdrücken wird oft der Konjunktionsoperator ( $\wedge$ ) weggelassen.

# Der Shannon'sche Entwicklungssatz

Jede Boolesche Funktion  $f : \mathbb{B}^n \rightarrow \mathbb{B}$  kann durch eine Variable  $x_i$  und die entsprechenden Co-Faktoren dargestellt werden:

$$f(x_0, \dots, x_i, \dots, x_{n-1}) = \underbrace{x_i f(x_0, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_{n-1})}_{\text{Co-Faktor } f(x_i=1)} \vee \underbrace{\bar{x}_i f(x_0, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_{n-1})}_{\text{Co-Faktor } f(x_i=0)}$$

**Beispiel:** Entwicklung nach den Variablen  $a$  und  $b$

$$\begin{aligned} f(a, b, c) &= (a \vee b)(\bar{b} \vee c)(\bar{a} \vee \bar{c}) \\ &= a \underbrace{((\bar{b} \vee c)\bar{c})}_{f(a=1)} \vee \bar{a} \underbrace{(b(\bar{b} \vee c))}_{f(a=0)} \\ &= a \underbrace{\left( b \underbrace{(0)}_{f(b=1)} \vee \bar{b} \underbrace{(\bar{c})}_{f(b=0)} \right)}_{f(a=1)} \vee \bar{a} \underbrace{\left( b \underbrace{(c)}_{f(b=1)} \vee \bar{b} \underbrace{(0)}_{f(b=0)} \right)}_{f(a=0)} \end{aligned}$$



# Das AND-OR-NOT-System

Jede Boolesche Funktion kann ausnahmslos durch alleinige Verwendung der Operatoren AND, OR und NOT dargestellt werden.

**Antivalenz:**  $a \oplus b = a\bar{b} \vee \bar{a}b$

**Äquivalenz:**  $a \Leftrightarrow b = \overline{a \oplus b} = ab \vee \bar{a}\bar{b}$

Ist eine Boolesche Funktion  $f : \mathbb{B}^n \rightarrow \mathbb{B}$  durch einen Booleschen Ausdruck gegeben, so liefert  $n$ -maliges Anwenden des Entwicklungssatzes einen Booleschen Ausdruck für  $f$  im AND-OR-NOT-System.

Operatoren, die nicht im AND-OR-NOT-System enthalten sind, werden durch Anwendung auf eine Konstante eliminiert.

$$x \Leftrightarrow 0 = \bar{x} \qquad x \Leftrightarrow 1 = x$$

$$x \oplus 0 = x \qquad x \oplus 1 = \bar{x}$$

$$x \Rightarrow 0 = \bar{x} \qquad x \Rightarrow 1 = 1$$

$$0 \Rightarrow x = 1 \qquad 1 \Rightarrow x = x$$

# Vollständige Operatorsysteme

Neben dem AND-OR-NOT-System gibt es noch andere so genannte vollständige Operatorsysteme. Vollständig weist auf die Darstellbarkeit jeder denkbaren Booleschen Funktion hin.

Zum Nachweis der Vollständigkeit eines Operatorsystems genügt es zu zeigen, dass die Grundfunktionen AND, OR und NOT darstellbar sind.

**Das NAND-System:**

$$\bar{a} = \overline{a \wedge a}$$
$$a \wedge b = \overline{\overline{a \wedge b}} = \overline{\overline{a \wedge b} \wedge \overline{a \wedge b}}$$
$$a \vee b = \overline{\overline{a \vee b}} = \overline{\overline{a} \wedge \overline{b}} = \overline{\overline{a \wedge a} \wedge \overline{b \wedge b}}$$

**Das NOR-System:**

$$\bar{a} = \overline{a \vee a}$$
$$a \vee b = \overline{\overline{a \vee b}} = \overline{\overline{a \vee b} \vee \overline{a \vee b}}$$
$$a \wedge b = \overline{\overline{a \wedge b}} = \overline{\overline{a} \vee \overline{b}} = \overline{\overline{a \vee a} \vee \overline{b \vee b}}$$

# 3.3 Darstellung Boolescher Funktionen

2.3.1 KV-Diagramme

2.3.2 Schaltpläne

2.3.3 Zweistufige Normalformen

2.3.4 Vektordarstellung

2.3.5 Binäre Entscheidungsgraphen

# KV-Diagramme

Karnaugh-Veitch-Diagramme sind spezielle Venn-Diagramme (bekannt aus der Mengenalgebra), die zur Darstellung und insbesondere auch zur Minimierung Boolescher Funktionen verwendet werden, wenn die Zahl der Variablen nicht größer als 4 ist.

Ein KV-Diagramm für eine Boolesche Funktion ist eine quadratische oder eine rechteckige (bestehend aus zwei Quadraten) Tafel, die aus Feldern zusammengesetzt wird, in die die Funktionswerte der Booleschen Funktion eingetragen werden.

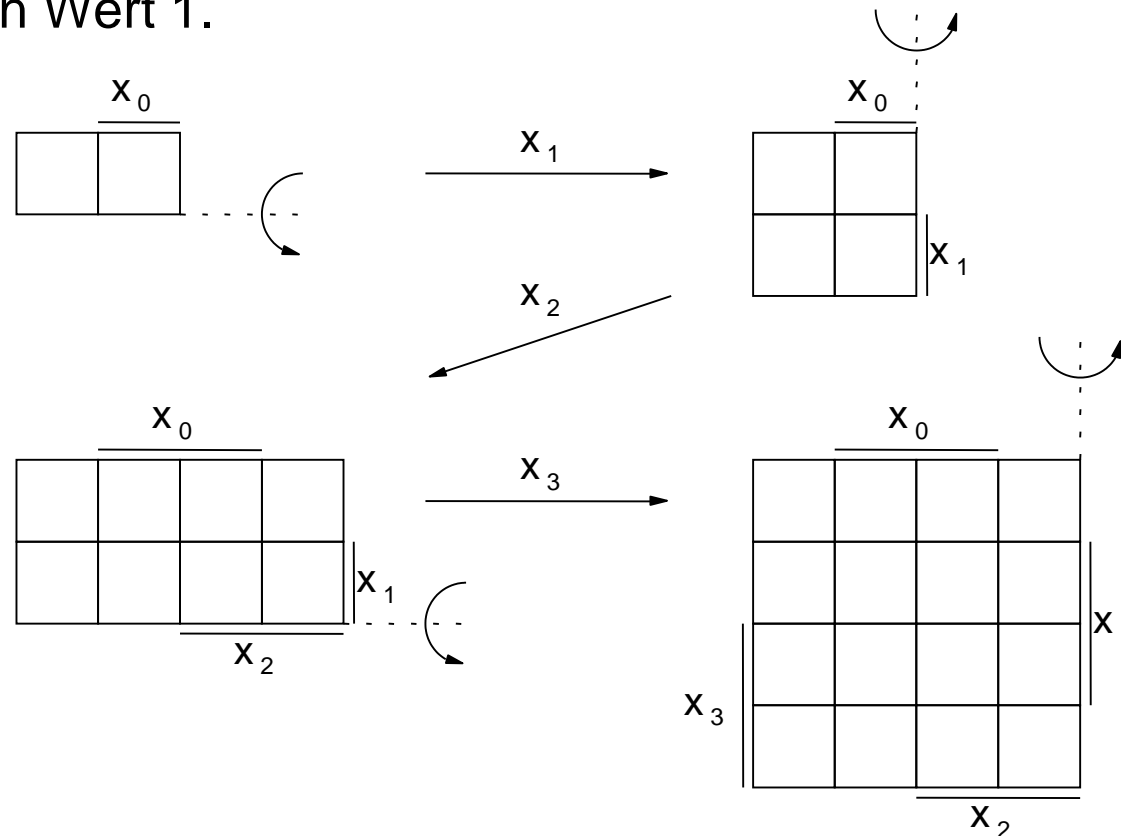
## Definition:

Ein KV-Diagramm von  $f : \mathbb{B}^n \rightarrow \mathbb{B}$  ist eine graphische Darstellung der Funktionstabelle von  $f$ , wobei jeder Zeile der Tabelle genau ein nummeriertes Feld im KV-Diagramm entspricht. Die Feldnummer entspricht der Zeilennummer, wenn die Variablen mit fallendem Index notiert werden.

KV-Diagramme sind ein Ersatz für die Funktionstabelle einer Booleschen Funktion.

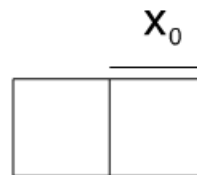
# Konstruktion von KV-Diagrammen

Ein KV-Diagramm für eine Funktion in  $n$  Variablen entsteht induktiv durch eine Verdopplung einer KV-Tafel für  $n-1$  Variablen durch Spiegelung. Die jeweils neu hinzukommende Variable hat in der alten Tafel den Wert 0, in der neuen den Wert 1.



# Konstruktion von KV-Diagrammen

Bei 5 Variablen muss man sich auch benachbarte Felder in der 3. Dimension vorstellen, daher sind die Diagramme mit 5 Variablen zwar möglich aber nicht mehr sehr anschaulich.



# KV-Diagramm für die 2-von-3 Funktion

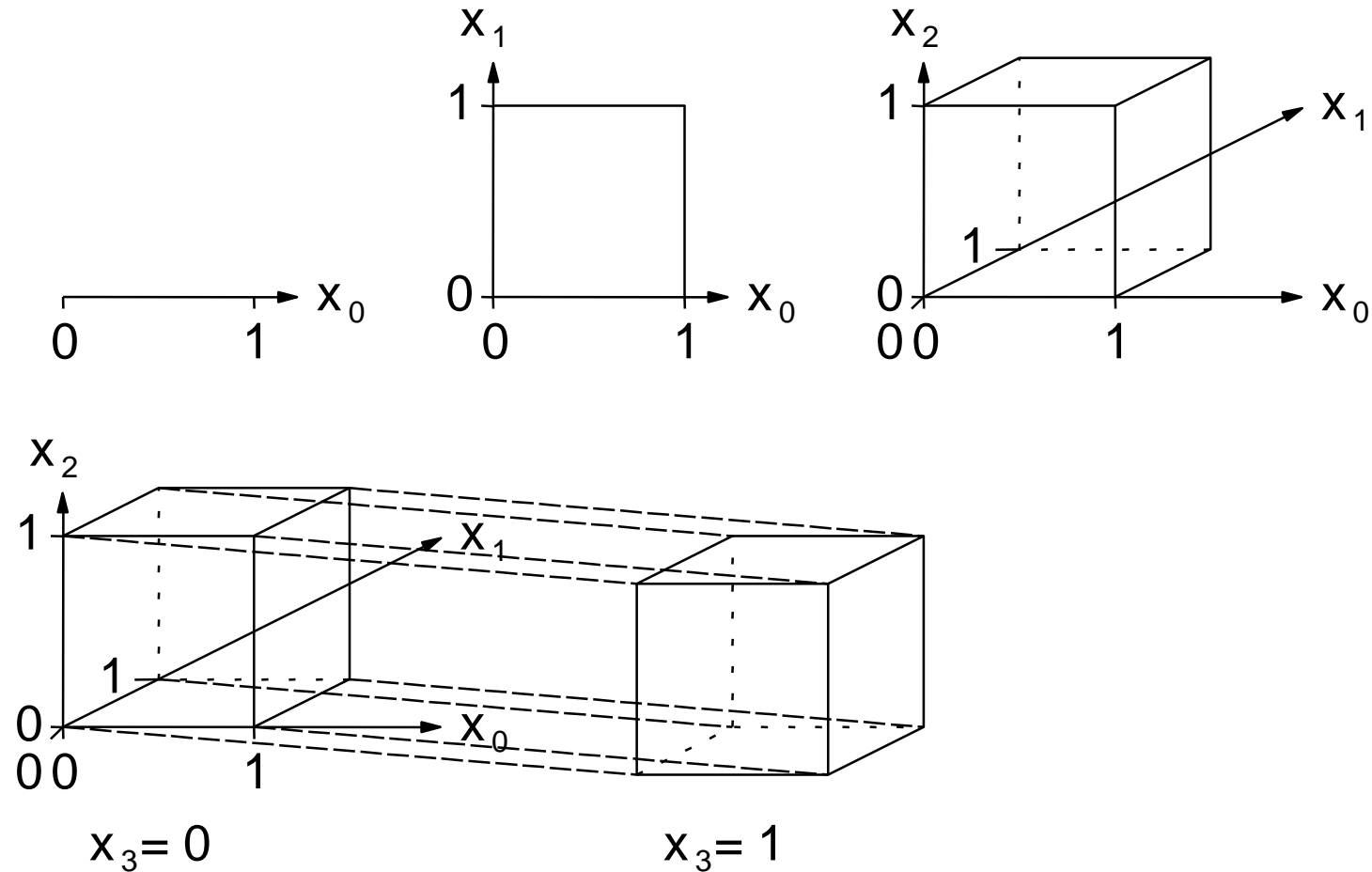
$x_2$	$x_1$	$x_0$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$x_0$				
0	0	1	0	$x_1$
0	1	1	1	
$x_2$				

$x_0$				
0	0	1	0	$x_1$
0	1	1	1	
$x_2$				

$x_2 x_0$  (points to cell  $x_2=0, x_0=1$ )  
 $x_1 x_0$  (points to cell  $x_1=0, x_0=1$ )  
 $x_2 x_1$  (points to cell  $x_2=1, x_1=1$ )

# Die ersten 4 Einheitshyperwürfel

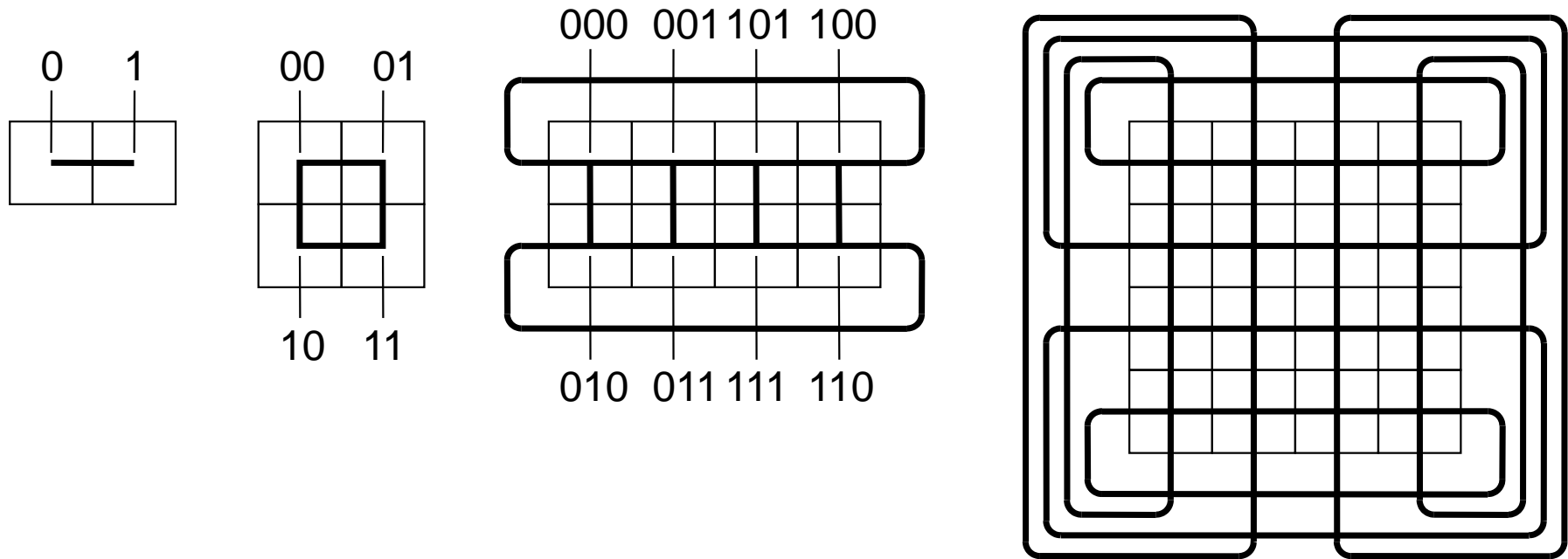




# Einheitshyperwürfel

Die KV-Diagramme für Schaltfunktionen von  $k$  Variablen  $k \in \{1,2,3,4\}$  entsprechen umkehrbar eindeutig den Einheitshyperwürfeln der Dimension  $k$ .

Veranschaulicht wird dies durch eine planare Zeichnung der Hyperwürfel:

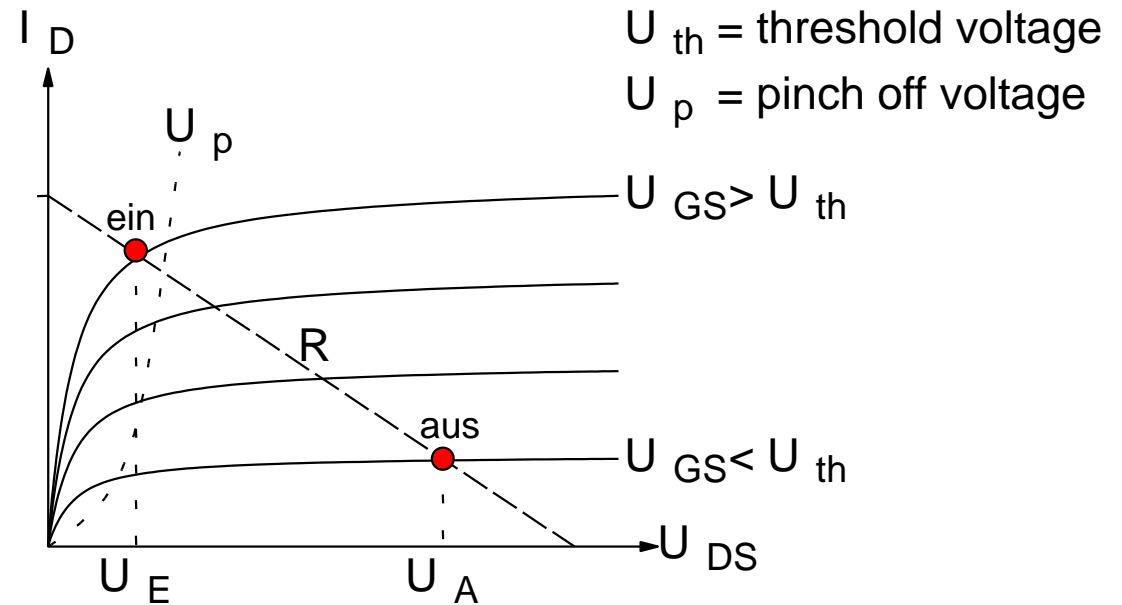
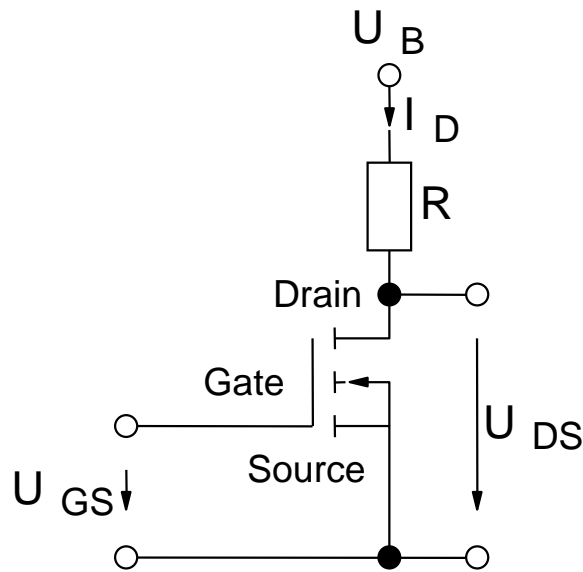


## 3.3.2 Schaltpläne

Schaltpläne sind gerichtete Graphen, die die Verknüpfungsoperatoren und die Klammerstruktur von Schaltfunktionen in einer in der Elektrotechnik üblichen genormten Art und Weise ausdrücken.

- Für die Knoten werden Gatterzeichen nach DIN verwendet.
- Sich kreuzende Kanten im Schaltplan haben in einer korrespondierenden physikalischen Realisierung genau dann eine (elektrische) Verbindung, wenn sie mit einem Punkt versehen sind.
- Negationen werden durch Kreise an den Gattereingängen oder Gatterausgängen repräsentiert.

# MOS-FET als Schalter



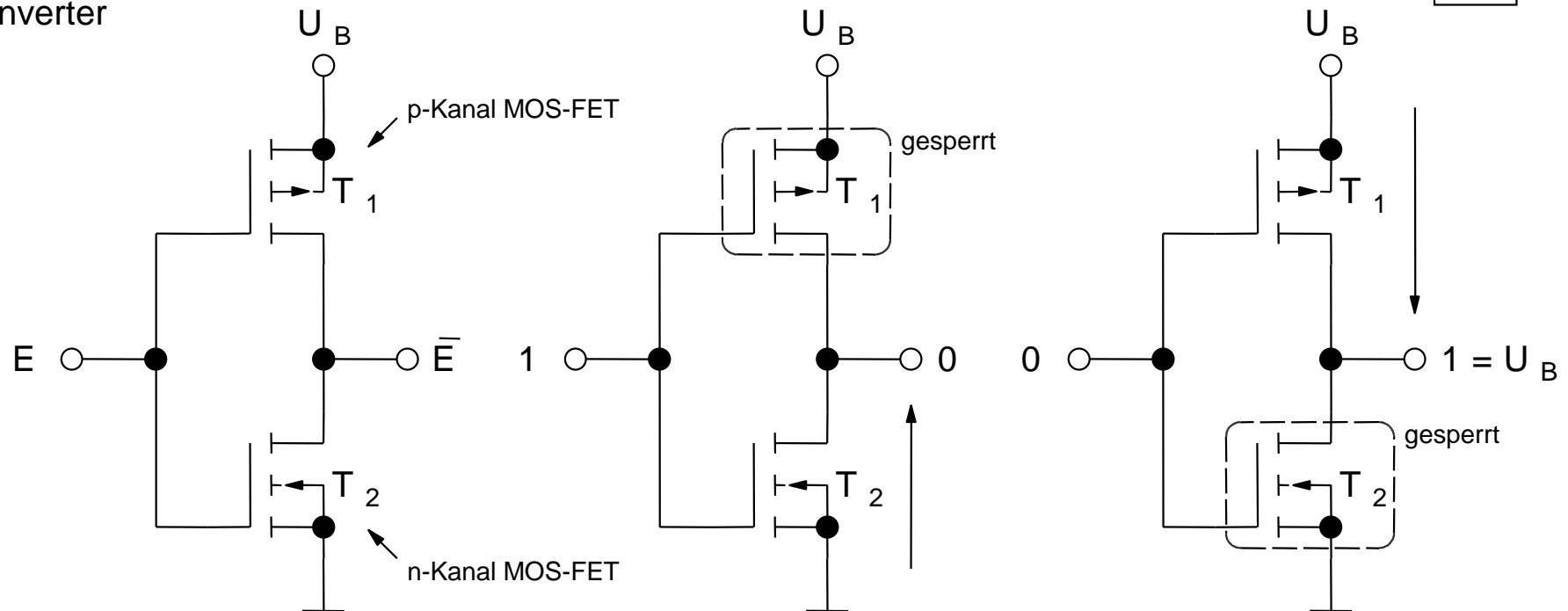
Die Schalterzustände 'ein'/'aus' werden durch die Zustände *Transistor aus* ( $U_{GS} < U_{th}$ ) und *Transistor ein* ( $U_{GS} > U_{th}$ ) mit dem Schnitt der Widerstandsgeraden realisiert.

# Verknüpfungsglieder: unipolare MOS-FETs

## Complementary MOS (CMOS) Technik (hier der Inverter):

Verknüpfungsglieder werden in dieser Technologie aus (selbstsperrenden) NMOS- und PMOS-Transistoren aufgebaut.

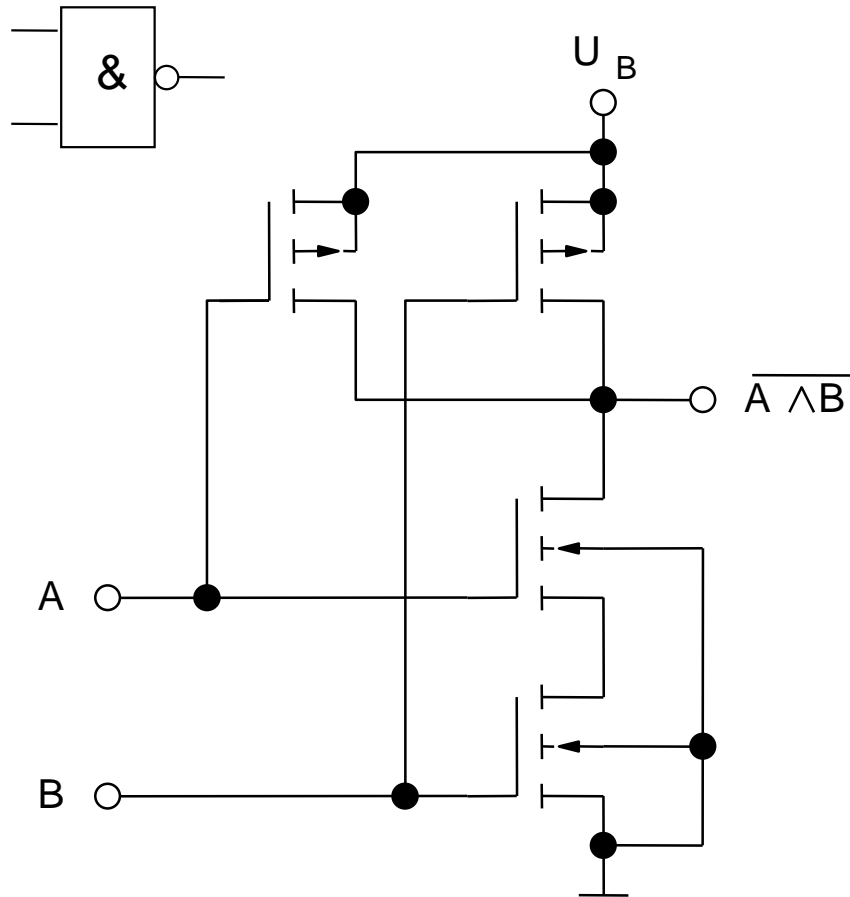
Inverter



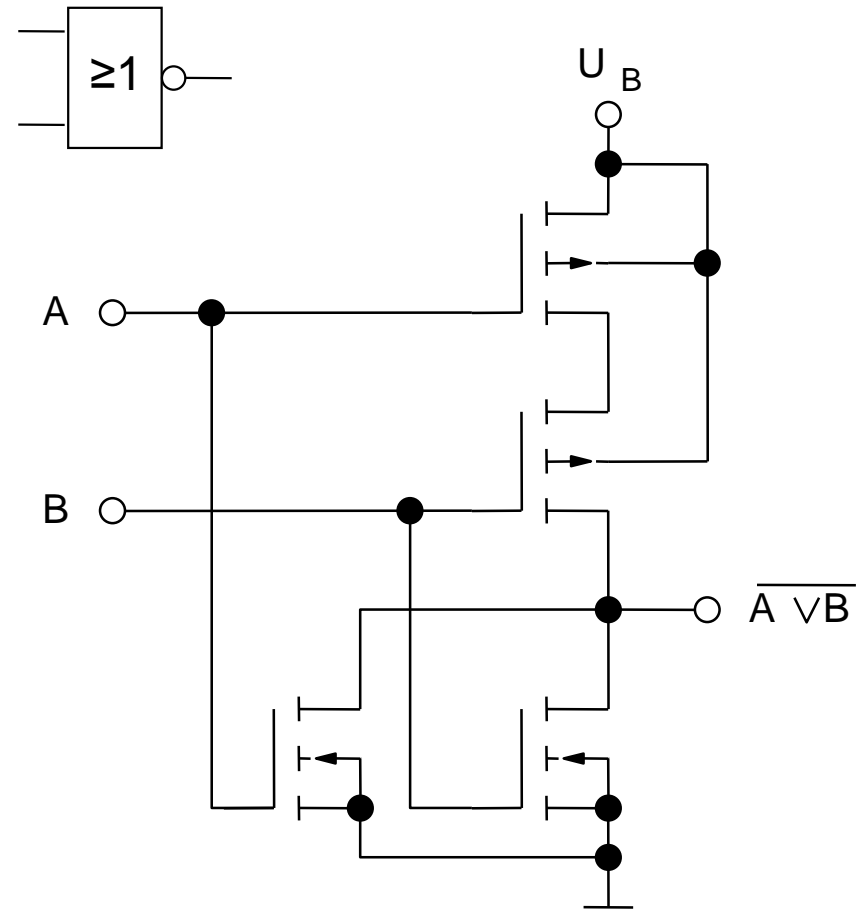
Es ist stets ein Transistor gesperrt und der andere leitend. Daher ist der Betriebsstrom und die statische Verlustleistung nahezu Null.

# Verknüpfungsglieder: MOS-FETs

NAND



NOR



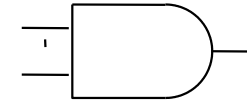
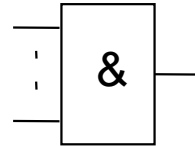
# Gattertypen

Gatter-Typ

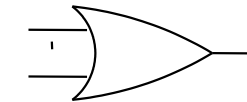
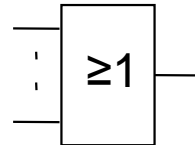
DIN/IEC 60617-12

USA

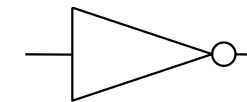
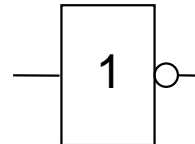
AND



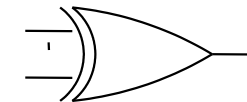
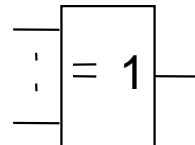
OR



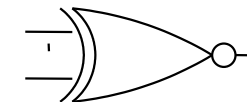
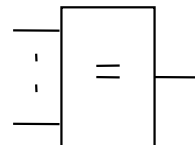
NOT



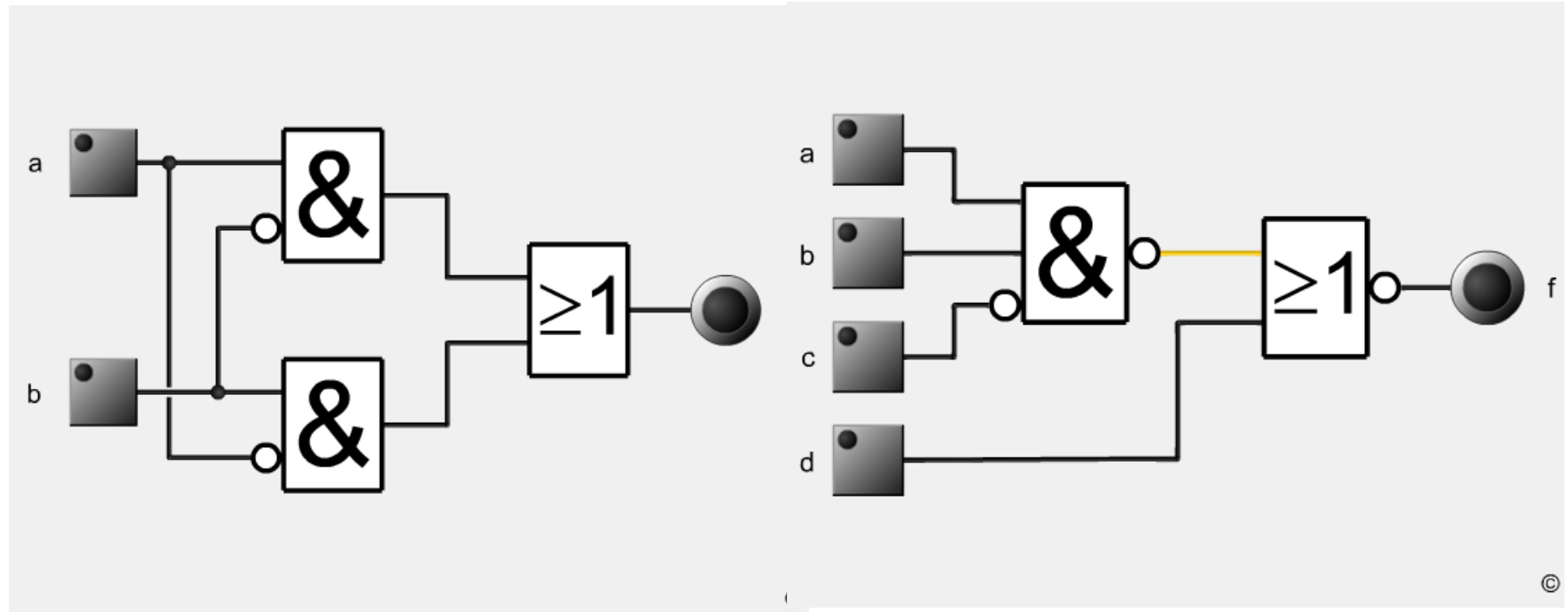
Antivalenz (XOR)



Äquivalenz (XNOR)



# Beispiele für einfache Schaltpläne

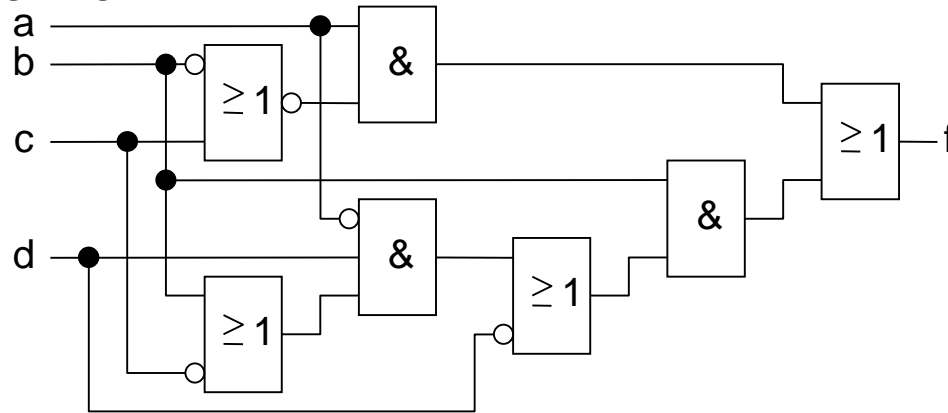


$$f = a \oplus b = \bar{a}b \vee a\bar{b}$$

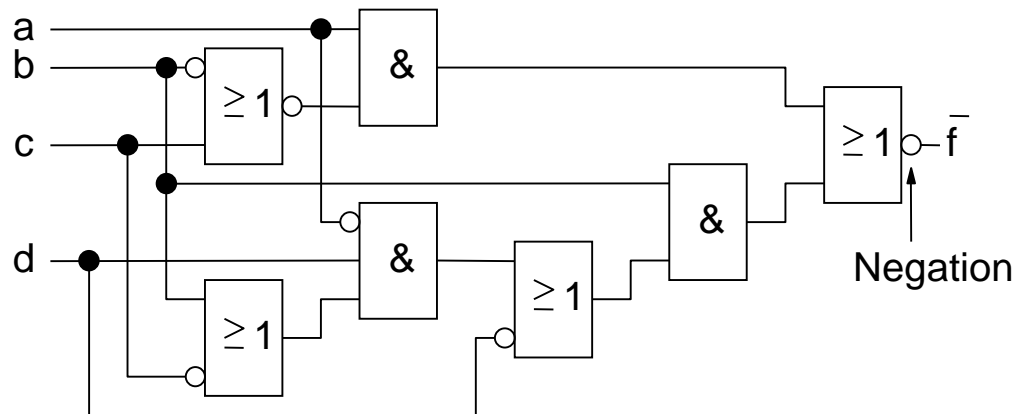
$$f = \overline{abc} \oplus d$$

# Graphische Negation von Schaltfunktionen

Ausgangsfunktion:



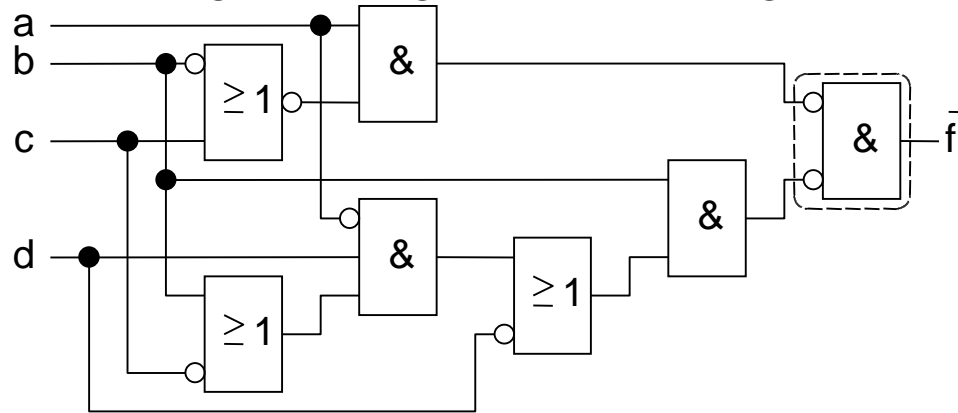
## 1. Negation des Ausgangs



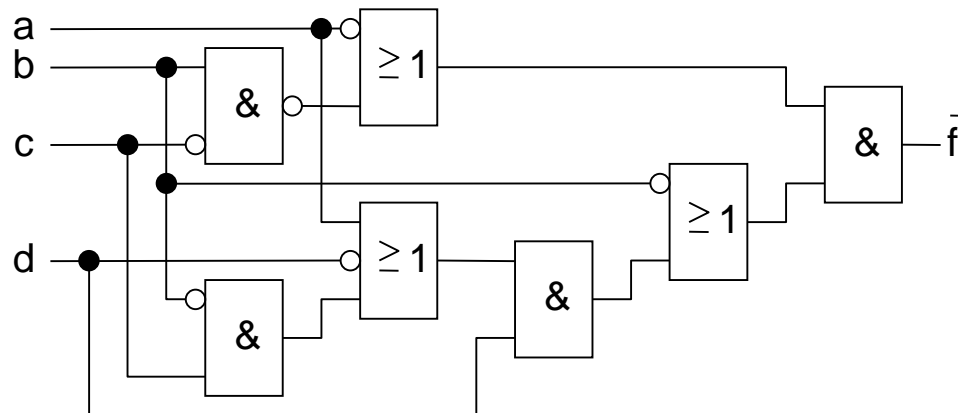


# Graphische Negation von Schaltfunktionen

## 2. Anwendung der Regel von DeMorgan



## Negierte Funktion



# Konjunktionsterme

## Konjunktionsterm:

Ein Konjunktionsterm ist eine Konjunktion von Literalen.

$$T^K = \bigwedge_{i \in \{0, \dots, n-1\}} \underline{x_i} \quad \underline{x_i} \in \{x_i, \overline{x_i}\}$$

Jede Variable tritt höchstens einmal auf.

## Minterm:

Ein Minterm  $M_i$  ist ein Konjunktionsterm maximaler Länge, d.h. es kommen sämtliche Literale genau einmal in  $M_i$  vor.

Es existiert *genau eine* Belegung der Literale in  $M_i$ , mit der  $M_i$  den Wert 1 annimmt.

## Minterm einer Schaltfunktion:

Ein Term  $M_i$  ist Minterm einer Booleschen Funktion  $f$ , wenn  $f$  für die Belegung den Wert 1 annimmt, für die auch  $M_i$  den Wert 1 annimmt.

# Minterme

Minterme einer Booleschen Funktion  $f$  entsprechen umkehrbar eindeutig

1. denjenigen Zeilen der Funktionstabelle, die als Funktionswert die 1 haben.
2. denjenigen Quadraten des KV-Diagramms dieser Funktion, die den Wert 1 tragen

## Bemerkung:

- Minterme werden mit fallenden Indizes notiert
- die duale Interpretation der 1-Belegung eines Minterms entspricht exakt der Zeilennummer seiner Position in der Funktionstabelle

## Beispiel:

$$x_0 x_1 \overline{x_2} \overline{x_3} \rightarrow \overline{x_3} \overline{x_2} x_1 x_0 \rightarrow 0011 \rightarrow M_3$$

# Disjunktionsterme

## Disjunktionsterm:

Ein Disjunktionsterm entsteht durch die disjunktive Verknüpfung von Literalen.

$$T^D = \bigvee_{i \in \{0, \dots, n-1\}} \underline{x_i} \quad \underline{x_i} \in \{x_i, \overline{x_i}\}$$

Jede Variable tritt höchstens einmal auf.

## Maxterm:

Ein Maxterm ist ein Disjunktionsterm maximaler Länge (analog Minterm). Im KV-Diagramm stellt sich der Maxterm als genau eine 0 dar. Die restlichen Felder sind mit 1 belegt (invers zum KV-Diagramm eines Minterms).

# Maxterme

Beispiel:

$$T^D = \overline{x_0} \vee \overline{x_1} \vee x_2 \vee x_3 = \overline{x_0 x_1 \overline{x_2} \overline{x_3}}$$

	$x_0$				
		1	1	1	1
		1	0	1	1
		1	1	1	1
		1	1	1	1
		$x_2$			
$x_3$				$x_1$	

# Implikant und Primimplikant

## Implikant

Ein Implikant  $I$  von  $f$  ist ein Konjunktionsterm, der  $f$  impliziert.

$$I = 1 \Rightarrow f = 1$$

Für jede Belegung, bei der  $I$  den Wert 1 annimmt, nimmt auch  $f$  den Wert 1 an.

## Primimplikant

Ein Primimplikant ist ein Implikant von  $f$ , der nicht mehr verkürzbar ist. Eine systematische Methode zur Bestimmung aller Primimplikanten muss unabhängig davon sein,

- in welcher Form die Boolesche Funktion angegeben ist und
- in welcher Reihenfolge die Booleschen Rechenregeln zur Minimierung der Funktion angewendet werden.

# Algorithmus zur Verkürzung eines Implikanten

Die Prüfung, ob ein Implikant  $I$  verkürzbar ist, läuft wie folgt ab:

**für** alle Literale  $\underline{x}_k$  in  $I$

$I$  um das Literal  $\underline{x}_k$  verkürzen zu  $I_k$

Restliche Literale  $\underline{x}_i$  von  $I_k$  mit 1 bzw. 0 belegen, so dass gilt:  $I_k = 1$

**Wenn** für alle nicht belegten Literale gilt:  $f = 1$ , **dann**

ist  $I_k$  ein Implikant der Funktion und  $I$  kein Primimplikant

**ende**

Umkehrschluss:

Wenn ein Implikant nach obigem Algorithmus nicht verkürzbar ist, ist er Primimplikant.

# Beispiel – Primimplikanten (1)

Funktion:

$$f(a, b, c, d) = ab \vee \bar{a}d \vee a\bar{b}d$$

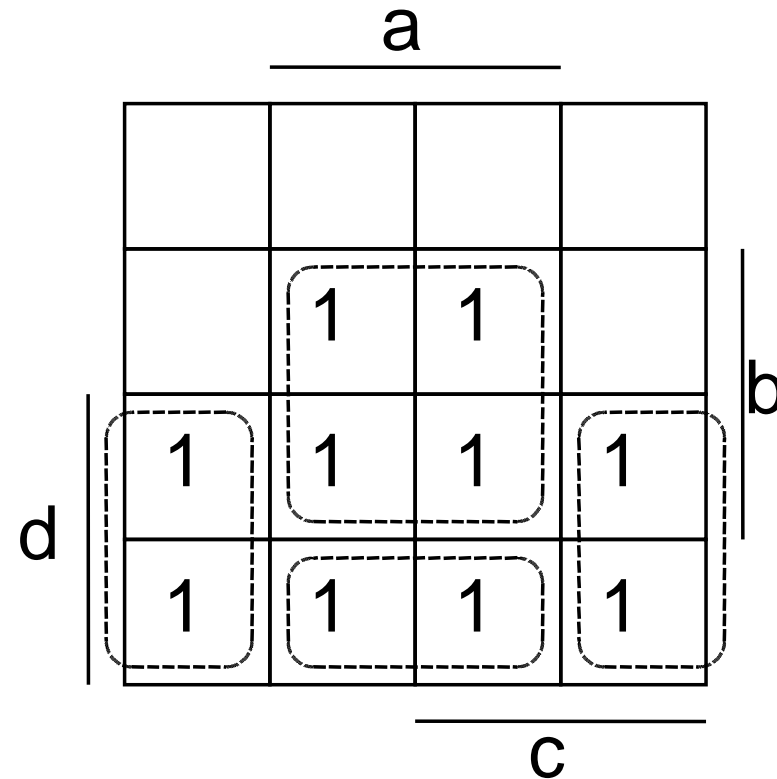
Implikanten:

$$I_1 = ab$$

$$I_2 = \bar{a}d$$

$$I_3 = a\bar{b}d$$

KV-Diagramm:





# Beispiel – Primimplikanten (2)

## Verkürzung des Implikanten $I_1 = ab$ :

Verkürze  $ab$  um  $\underline{a}$ :  $I_1^a = b \Rightarrow f(a, 1, c, d) = a \vee \bar{a}d \neq 1$

Verkürze  $ab$  um  $\underline{b}$ :  $I_1^b = a \Rightarrow f(1, b, c, d) = b \vee \bar{b}d \neq 1$

Primimplikant:  $ab$

## Verkürzung des Implikanten $I_2 = \bar{a}d$ :

$$I_2^a = d \Rightarrow f(a, b, c, 1) = ab \vee \bar{a} \vee a\bar{b} = 1$$

$$I_2^d = \bar{a} \Rightarrow f(0, b, c, d) = d \neq 1$$

verkürzter Implikant:  $d \Rightarrow \bar{a}d$  ist kein Primimplikant

# Beispiel – Primimplikanten (3)

Verkürzung des Implikanten  $I_3 = a\bar{b}d$ :

$$I_3^a = \bar{b}d \Rightarrow f(a, 0, c, 1) = \bar{a} \vee a = 1$$

$$I_3^{a,b} = d \Rightarrow f(a, b, c, 1) = ab \vee \bar{a} \vee a\bar{b} = 1$$

$$I_3^{a,d} = \bar{b} \Rightarrow f(a, 0, c, d) = \bar{a}d \vee ad \neq 1$$

$$I_3^b = ad \Rightarrow f(1, b, c, 1) = b \vee \bar{b} = 1$$

$$I_3^{b,a} = d \Rightarrow f(a, b, c, 1) = ab \vee \bar{a} \vee a\bar{b} = 1$$

$$I_3^{b,d} = a \Rightarrow f(1, b, c, d) = b \vee \bar{b}d \neq 1$$

$$I_3^d = \bar{a}\bar{b} \Rightarrow f(1, 0, c, d) = d \neq 1$$

verkürzter Implikant:  $d \Rightarrow a\bar{b}d$  ist kein Primimplikant

# Beispiel – Primimplikanten (4)

## Schlussfolgerung:

Bei dem Implikanten  $I = d$  handelt es sich um einen Primimplikanten, weil er nicht verkürzt werden kann:

$$I^d = 1 \Rightarrow f(a, b, c, d) = ab \vee \bar{a}d \vee a\bar{b}d \neq 1$$

Die Implikanten  $I_2$  und  $I_3$  können durch den Primimplikanten  $I = d$  ersetzt werden. Für die Funktion  $f$  gilt:

$$f(a, b, c, d) = I_1 \vee I = ab \vee d$$

# Beispiel – Primimplikanten (5)

Funktion:

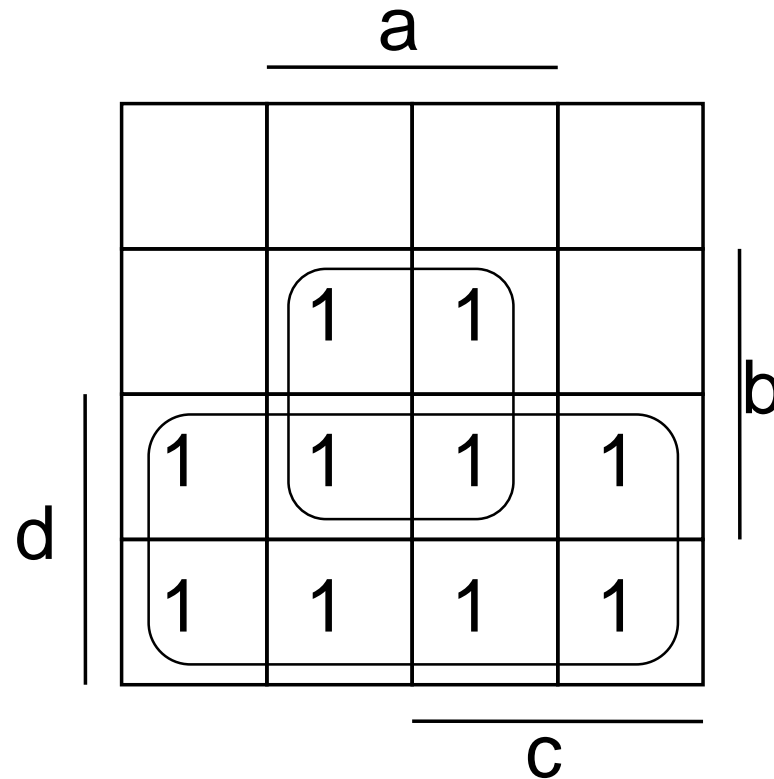
$$f(a, b, c, d) = ab \vee d$$

Primimplikanten:

$$I_1 = ab$$

$$I_2 = d$$

KV-Diagramm:



## 3.3.3 Zweistufige Normalformen

### Disjunktive Normalform (DNF):

Die DNF einer Schaltfunktion  $f$  ist die Disjunktion von Konjunktionstermen, die Implikanten dieser Funktion sind.

$$f(X) = \bigvee_{i=0}^{l-1} T_i^K \qquad T_i^K = \bigwedge_{\forall j \in N_i} x_{ij}$$

### Konjunktive Normalform (KNF):

Ein Boolescher Ausdruck liegt genau dann in konjunktiver Normalform vor, wenn er eine Konjunktion von Disjunktionstermen darstellt.

$$f(X) = \bigwedge_{i=0}^{l-1} T_i^D \qquad T_i^D = \bigvee_{\forall j \in N_i} x_{ij}$$

# Kanonische Normalformen

Die zweistufige Normalform einer Funktion ist nicht eindeutig bestimmt.

**Beispiel:** Disjunktive Normalformen der 2-von-3 Funktion

$$\begin{aligned} f &= x_1 x_0 \vee x_2 x_0 \vee x_2 x_1 \\ &= x_2 x_1 x_0 \vee x_2 x_1 \overline{x_0} \vee x_2 \overline{x_1} x_0 \vee \overline{x_2} x_1 x_0 \end{aligned}$$

## Kanonische disjunktive Normalform (KDNF):

Eine zweistufige disjunktive Normalform (DNF), die nur Minterme enthält, heißt kanonische disjunktive Normalform.

## Kanonische konjunktive Normalform (KKNF):

Eine zweistufige konjunktive Normalform (KNF), die nur Maxterme enthält, heißt kanonische konjunktive Normalform.

Die kanonischen Normalformen einer Booleschen Funktion sind, bis auf die Permutation der Terme und Literale innerhalb der Terme, eindeutig bestimmt.

# Kanonische DNF

Die KDNF einer Booleschen Funktion lässt sich direkt aus der Funktionstabelle ableiten.

**Beispiel:** Die 2-von-3 Funktion

$x_2$	$x_1$	$x_0$	$f$	$M_i$
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{x_2}x_1x_0$
1	0	0	0	
1	0	1	1	$x_2\overline{x_1}x_0$
1	1	0	1	$x_2x_1\overline{x_0}$
1	1	1	1	$x_2x_1x_0$

$$f = x_2x_1x_0 \vee x_2x_1\overline{x_0} \vee x_2\overline{x_1}x_0 \vee \overline{x_2}x_1x_0$$

# Konjunktive Normalform

Die KNF einer Funktion  $f$  lässt sich leicht aus der DNF von  $\overline{f}$  ableiten. Hierzu ist lediglich der Shannon'sche Inversionssatz anzuwenden, d.h. es müssen alle Operatoren vertauscht und die Literale negiert werden.

**Beispiel:** Die 2-von-3 Funktion

$$\overline{f} = \overline{x_1} \overline{x_0} \vee \overline{x_2} \overline{x_0} \vee \overline{x_2} \overline{x_1}$$

$\Downarrow$

$$f = (x_1 \vee x_0)(x_2 \vee x_0)(x_2 \vee x_1)$$

Um  $\overline{f}$  in DNF zu erhalten kann man von  $f$  (z.B. in DNF) ausgehen, den Shannon'schen Inversionssatz anwenden, ausmultiplizieren und damit wieder in eine DNF umwandeln.

In der *kanonischen* Konjunktiven Normalform (KKNF) werden analog zur KDNF ausschließlich Maxterme als Disjunktionsterme verwendet.

**Beispiel:** Die 2-von-3 Funktion

$$f = (x_2 \vee x_1 \vee x_0)(x_2 \vee x_1 \vee \overline{x_0})(x_2 \vee \overline{x_1} \vee x_0)(\overline{x_2} \vee x_1 \vee x_0)$$



# Vergleich boolescher Funktionen

Eine **kanonische Normalform** ermöglicht den direkten Vergleich auf Gleichheit zweier boolescher Funktionen. Dies kann verwendet werden um

- die **Äquivalenz** zweier Gatterschaltungen zu zeigen

Vorgehen: 1. Forme die Gatterschaltungen in eine kanonische Form um

2. Ordne diese und zeige durch direkten Vergleich der Literale die Äquivalenz oder die Nichtäquivalenz.

- die Äquivalenz einer Gatterschaltung mit einer booleschen Funktion zu zeigen. Z.B. beschreibt die boolesche Funktion die Funktionsbeschreibung (Spezifikation) einer Schaltung und die Gatterschaltung ist die Implementierung

Problem: KKNF und KDNF sind unter Umständen sehr groß  
 $2^n$  (Min/Maxterme)

# Weitere Normalformen

In der theoretischen Informatik (z.B. DISMOD) wird auch noch die **Negationsnormalform** (NNF) eingeführt. Sie hat folgende Eigenschaften.

- Negationen kommen nur noch direkt bei den Literalen vor.
- Als Verknüpfungen kommen nur  $\wedge$  und  $\vee$  vor.
- Sie ist jedoch nicht 2-stufig sondern kann mehrstufig sein.
- Sie ist nicht kanonisch.

## 3.3.4 Vektordarstellung

Die Größe

$$X = [x_i] = [x_0 \ x_1 \ \dots \ x_{n-1}]$$

heißt Vektor mit den Elementen oder Komponenten  $x_i$ .

Eine Funktion

$$y = f(x_0, \dots, x_{n-1})$$

lässt sich durch

$$y = f(X)$$

Darstellen.

# Vektoroperationen

**Negation:**

$$\overline{A} = \overline{[a_i]} = [\overline{a_i}]$$

**Zweistellige Operationen:**

$$A \circ B = [a_i] \circ [b_i] = [a_i \circ b_i]$$

Das Symbol  $\circ$  steht stellvertretend für einen beliebigen Operator.

**Vektorreduktion:**

$$\circ/A = \circ/[a_i] = (\dots((a_0 \circ a_1) \circ a_2) \circ \dots \circ a_{n-1})$$

**Beispiel:**

$$\wedge/A = \bigwedge_{i=0}^{n-1} a_i$$

# Beispiel – Äquivalenz zweier Binärzahlen

Mit Hilfe der gezeigten Formalismen kann die Äquivalenz zweier Binärzahlen wie folgt geschrieben werden:

$$g = f(X, Y) = \bigwedge_{i=0}^{n-1} (x_i \Leftrightarrow y_i)$$

$$G = X \Leftrightarrow Y = [x_0 \Leftrightarrow y_0 \ x_1 \Leftrightarrow y_1 \ \dots \ x_{n-1} \Leftrightarrow y_{n-1}]$$

$$g = \wedge/G = g_0 \wedge g_1 \wedge \dots \wedge g_{n-1}$$

$$g = \wedge/(X \Leftrightarrow Y)$$

# Matrixoperationen

**Matrix:**

$$A_M = [a_{ij}] = \begin{bmatrix} a_{0\ 0} & a_{0\ 1} & \cdots & a_{0\ m-1} \\ a_{1\ 0} & & & \vdots \\ \vdots & & & \vdots \\ a_{r-1\ 0} & \cdots & \cdots & a_{r-1\ m-1} \end{bmatrix}$$

**Transposition:**

$$A_M^T = [a_{ij}]^T = [a_{ji}]$$

**Negation:**

$$\overline{A_M} = \overline{[a_{ij}]} = [\overline{a_{ij}}]$$

**Zweistellige Operationen:**

$$A_M \otimes B_M = [a_{ij}] \otimes [b_{ij}] = [a_{ij} \otimes b_{ij}]$$

# Matrixoperationen

## Verallgemeinerte Multiplikation:

$$A_M \oplus \otimes B_M = [A_i] \oplus \otimes [B_j] = [\oplus / (A_i \otimes B_j)]$$

Die Symbole  $\oplus$  und  $\otimes$  stehen für beliebige Matrixoperationen.

Zur Darstellung einer Vektorfunktion (Bündelfunktion)  $Y = f(X)$  in kanonischer Form werden die folgenden Matrixoperationen definiert:

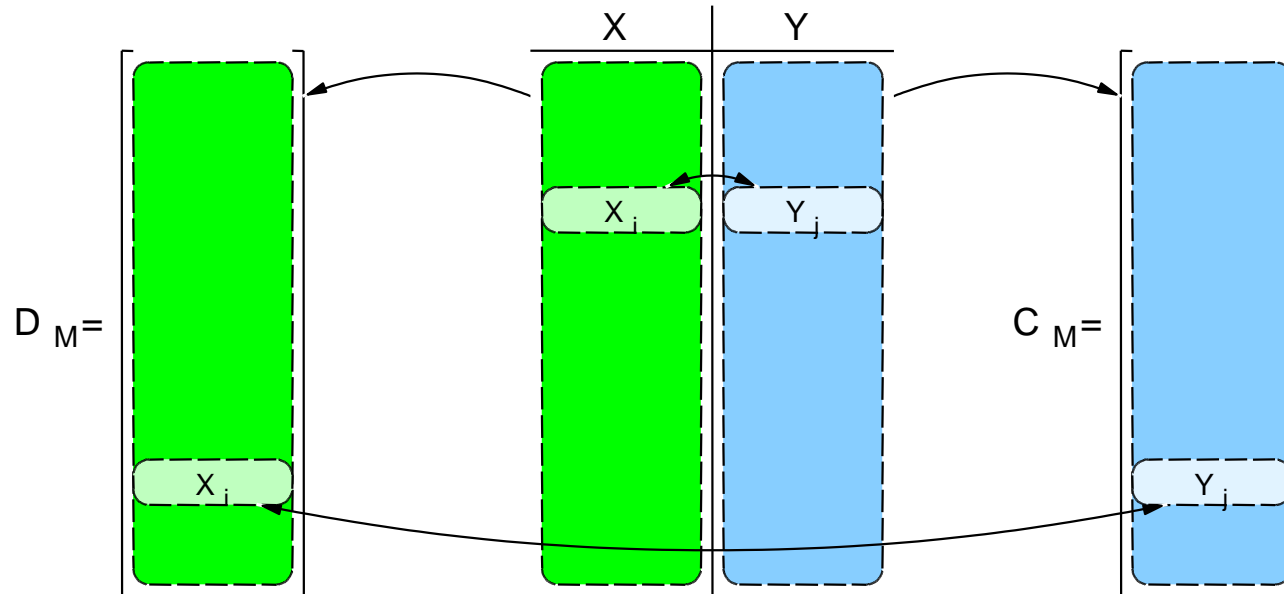
- Die Elemente  $y_{ij}$  von  $A_M \vee \wedge B_M$  bzw.  $A_M \wedge \vee B_M$  entstehen aus der Konjunktion  $A_i \wedge B_j$  bzw. der Disjunktion  $A_i \vee B_j$  der Spalten- ( $B_j$ ) und Zeilenvektoren ( $A_i$ ):

$$A_M \vee \wedge B_M = [\vee / (A_i \wedge B_j)]$$

$$A_M \wedge \vee B_M = [\wedge / (A_i \vee B_j)]$$

# Vektordarstellung

Bei der Vektordarstellung werden Boolesche Funktionen durch Matrizen spezifiziert.



Die Zeilen der Matrizen können beliebig permutiert sein, jedoch muss die Zuordnung der Eingangsbelegungen zu den Funktionswerten erhalten bleiben.

$$Y = (X \wedge \Leftrightarrow D_M^T) \vee \wedge C_M$$



# Beispiel – Halbaddierer

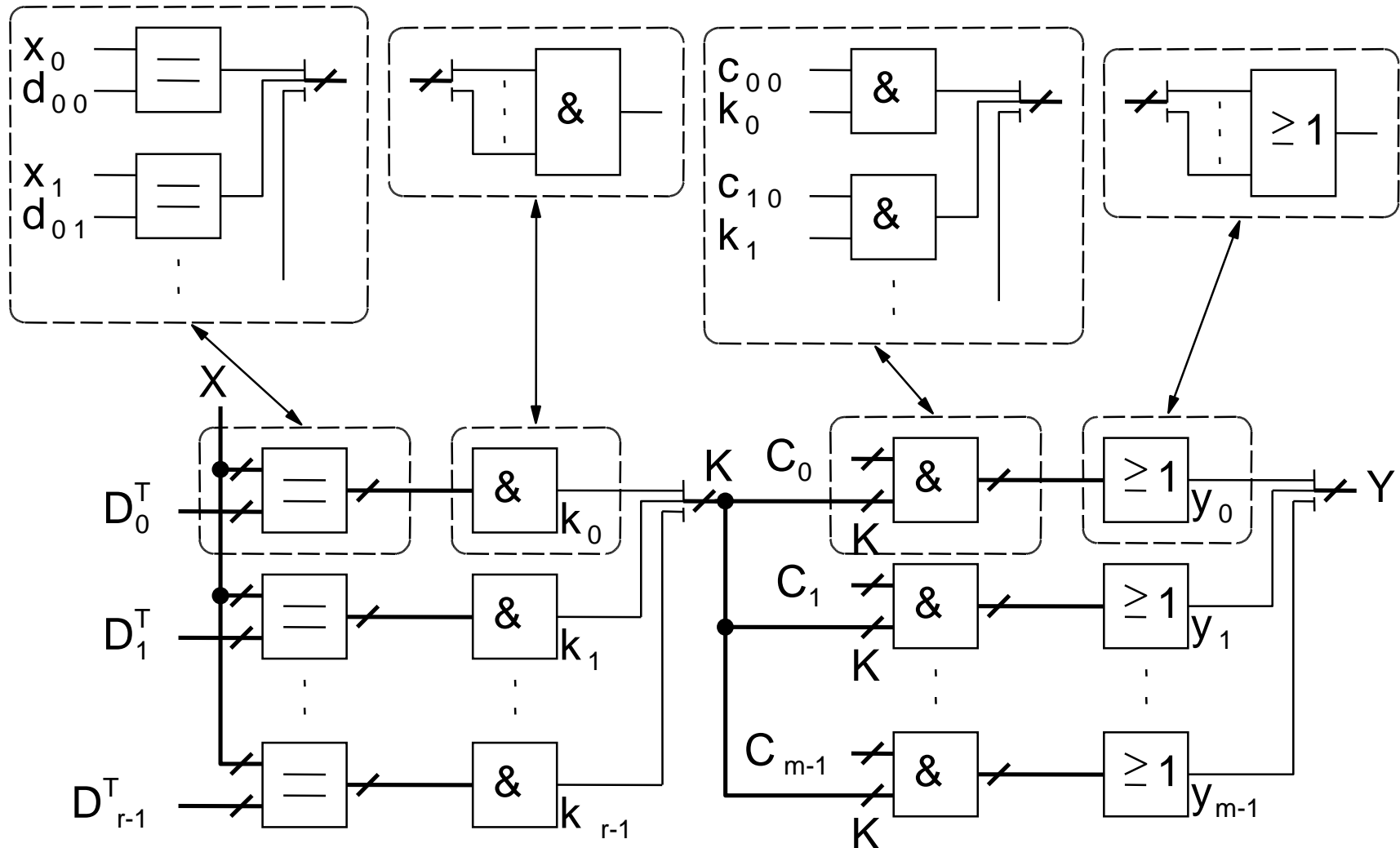
$$D_M = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \begin{array}{c|cc} x_0 & x_1 & y_0 (C_{i+1}) & y_1 (S_i) \\ \hline 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array} \quad C_M = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\underbrace{[y_0 \ y_1]}_Y = (\underbrace{[x_0 \ x_1]}_X \wedge \Leftrightarrow D_M^T) \vee \wedge C_M = \underbrace{[x_0 \ x_1] \wedge \Leftrightarrow \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}}_{[x_0 \Leftrightarrow 0 \wedge x_1 \Leftrightarrow 0 = \overline{x_0} \overline{x_1} \quad \overline{x_0} x_1 \quad x_0 \overline{x_1} \quad x_0 x_1]} \vee \wedge \overbrace{\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}}^{\text{Codiermatrix } C_M}$$

$$y_0 = (\overline{x_0} \overline{x_1} \wedge 0) \vee (\overline{x_0} x_1 \wedge 0) \vee (x_0 \overline{x_1} \wedge 0) \vee (x_0 x_1 \wedge 1)$$

$$y_1 = (\overline{x_0} \overline{x_1} \wedge 0) \vee (\overline{x_0} x_1 \wedge 1) \vee (x_0 \overline{x_1} \wedge 1) \vee (x_0 x_1 \wedge 0)$$

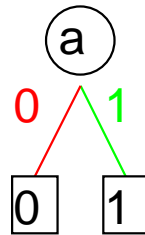
# Strukturdarstellung – KDNF



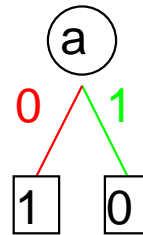
## 3.3.5 Binäre Entscheidungsgraphen

Bei Binären Entscheidungsgraphen (Binary Decision Diagrams (BDD)) handelt es sich um eine Datenstruktur zur kompakten Darstellung und günstigen algorithmischen Handhabung Boolescher Funktionen. Die Darstellung entspricht einem gerichteten, azyklischen Graphen, dem ein Entscheidungsprozeß zugrunde liegt.

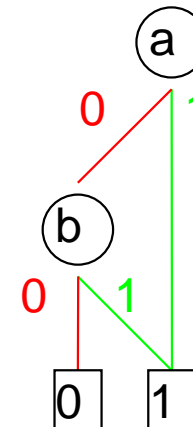
**Beispiele:**



$$f = a$$



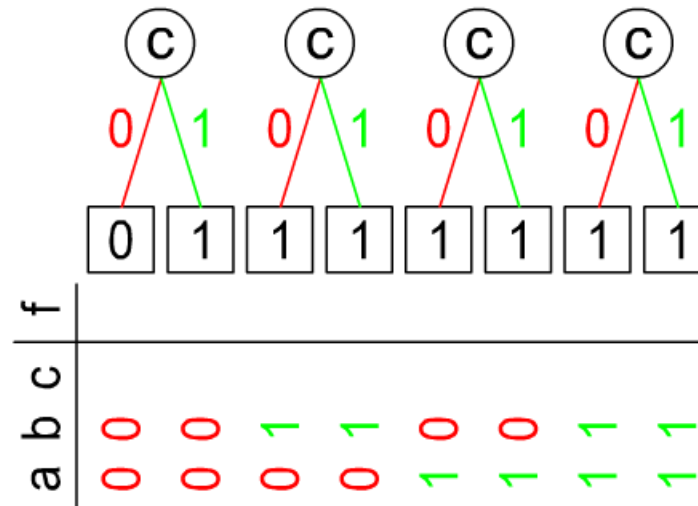
$$f = \bar{a}$$



$$f = a \vee b$$

# Konstruktion von BDDs – Funktionstafel

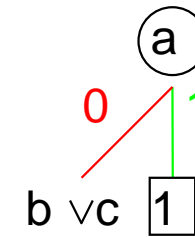
BDD der Funktion  $f(a, b, c) = a \vee b \vee c$ :



# Konstruktion von BDDs – Shannon'scher Entwicklungssatz (1)

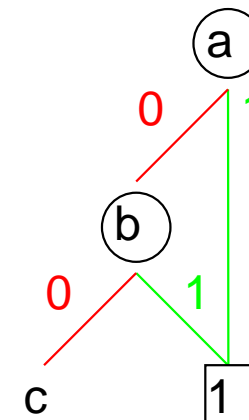
Entwicklung nach  $a$ :

$$\begin{aligned} f(a, b, c) &= a \vee b \vee c \\ &= a \underbrace{(1)}_{f(a=1)} \vee \bar{a} \underbrace{(b \vee c)}_{f(a=0)} \end{aligned}$$



Entwicklung nach  $b$ :

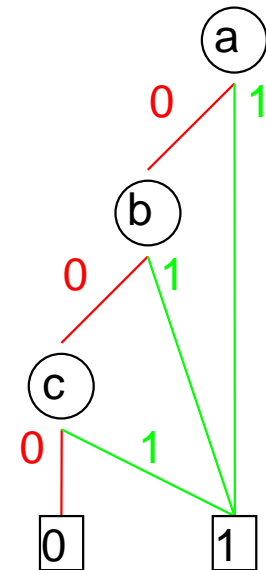
$$\begin{aligned} f(a, b, c) &= a \underbrace{(1)}_{f(a=1)} \vee \bar{a} \underbrace{(b \vee c)}_{f(a=0)} \\ &= a \underbrace{(1)}_{f(a=1)} \vee \bar{a} \underbrace{(b \underbrace{(1)}_{f(b=1)} \vee \bar{b} \underbrace{(c)}_{f(b=0)})}_{f(a=0)} \end{aligned}$$



# Konstruktion von BDDs – Shannon'scher Entwicklungssatz (2)

Entwicklung nach  $c$ :

$$\begin{aligned} f(a, b, c) &= a \underbrace{(1)}_{f(a=1)} \vee \bar{a} \underbrace{(b \underbrace{(1)}_{f(b=1)} \vee \bar{b} \underbrace{(c)}_{f(b=0)})}_{f(a=0)} \\ &= a \underbrace{(1)}_{f(a=1)} \vee \bar{a} \underbrace{(b \underbrace{(1)}_{f(b=1)} \vee \bar{b} \underbrace{(c \underbrace{(1)}_{f(c=1)} \vee \bar{c} \underbrace{(0)}_{f(c=0)})}_{f(b=0)})}_{f(a=0)} \end{aligned}$$



## Bemerkung:

Die Teil-Graphen des BDD entsprechen den Co-Faktoren, die durch den Shannon'schen Entwicklungssatz entstehen.

# Weitere binäre Entscheidungsgraphen

## Sortierte BDDs (Ordered BDD (OBDD)):

Ein BDD heißt sortiert, wenn auf jedem Pfad die Reihenfolge der Variablen einer einheitlichen, festen Ordnung unterliegt.

### Bemerkung:

Für eine Boolesche Funktion  $f$  gibt es auch bei gegebener Ordnung  $\tau$  i. A. mehrere OBDDs, die nicht isomorph sind.

## Reduzierte OBDDs (ROBDDs)

Ein OBDD heißt reduziert, wenn die Knotenanzahl minimal ist.

### Bemerkungen:

Für eine gegebene Variablen-Ordnung  $\tau$  ist der reduzierte OBDD einer Funktion  $f$  bis auf Isomorphismen eindeutig bestimmt und jeder andere BDD enthält mehr Knoten.

Die Knoten werden mit der Terminal-, der Eliminations- und der Isomorphieregel reduziert. Die Reduktion ist abgeschlossen, wenn keine dieser Regeln mehr angewendet werden kann.

# Reduktionsregeln – Terminalregel

Fasse sämtliche Terminalknoten (Blätter) mit gleichem Wert zusammen, so dass lediglich zwei Terminale mit den Werten 0 bzw. 1 im Graphen verbleiben. Ausgehend von den Terminalen werden Ebene für Ebene sukzessive die Eliminations- und die Isomorphieregel angewendet.

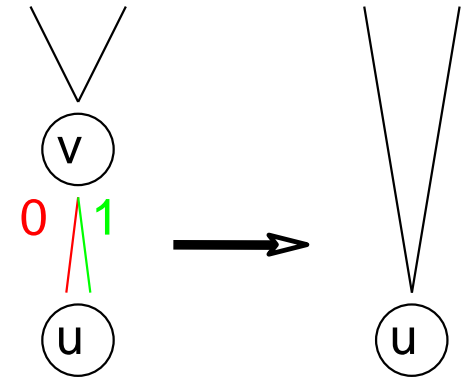
**Beispiel:** Die 2-von-3 Funktion



# Reduktionsregeln – Eliminationsregel

Wenn beide Kanten (0 und 1) eines gegebenen Knotens  $v$  zum gleichen Knoten  $u$  führen, dann eliminiere  $v$  und lenke alle eingehenden Kanten auf  $u$  um.

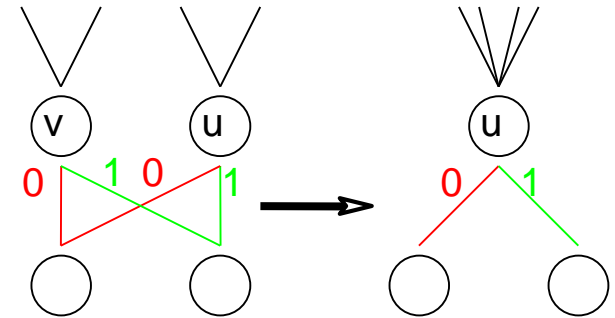
**Beispiel:** Die 2-von-3 Funktion



# Reduktionsregeln – Isomorphieregel

Wenn zu dem gerade durchlaufenen Nichtterminalknoten  $v$  ein bereits durchlaufener Knoten  $u$  mit gleichem Variablen-Index existiert, dessen Kanten beide jeweils zu den gleichen Knoten wie die von  $v$  führen, dann eliminiere  $v$  und lenke alle eingehenden Kanten zum verbleibenden Knoten  $u$  um.

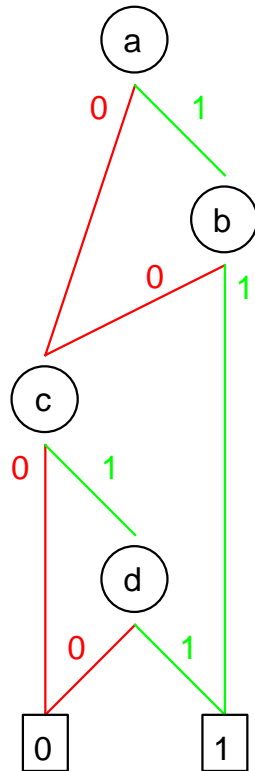
**Beispiel:** Die 2-von-3 Funktion



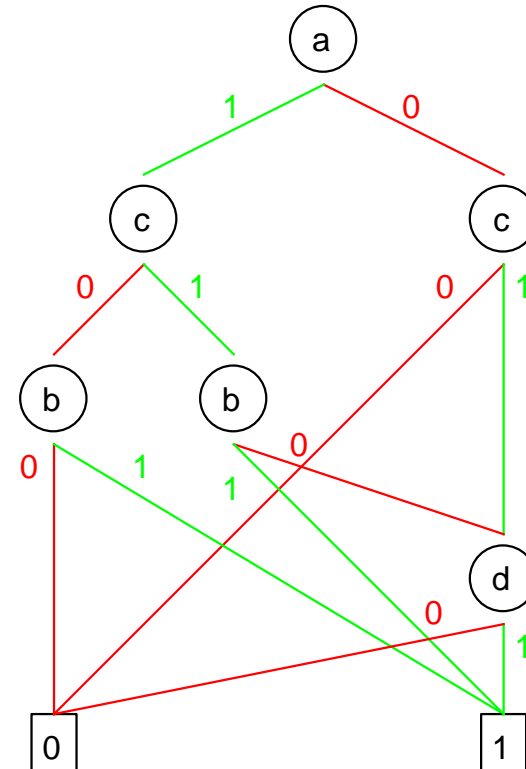
# Kosten der ROBDD-Darstellung

Die Knotenanzahl eines ROBDD ist von der Variablenordnung  $\tau$  abhängig.

**Beispiel:**  $f = ab \vee cd$



$\tau = a < b < c < d$



$\tau = a < c < b < d$

# Funktionstypen

Funktionen können bzgl. der OBDD-Darstellung in Funktionsklassen eingeteilt werden:

1. Die Größe des BDDs wächst unabhängig von der Variablenreihenfolge polynomial mit der Anzahl der Variablen.

**Beispiel:**  $f = x_0 \oplus x_1 \oplus \dots \oplus x_{n-1}$  (Paritätsfunktion)

Der BDD der Paritätsfunktion enthält  $2n + 1$  Knoten.

2. Die Größe des BDDs hängt stark von der Variablenreihenfolge ab.

**Beispiel:**  $f = x_0x_1 \vee x_2x_3 \vee \dots \vee x_{2n-2}x_{2n-1}$

Die Variablenordnung  $\tau = x_0 < x_1 < \dots < x_{2n-1}$  führt zu einer Darstellung mit  $2n+2$  Knoten, wohingegen die Variablenordnung  $\tau = x_0 < x_n < x_1 < x_{n+1} < \dots < x_{2n-1}$  zu einer Darstellung mit  $2^{n+1}$  Knoten führt.

3. Die Größe des BDDs wächst unabhängig von der Variablenreihenfolge exponentiell mit der Anzahl der Variablen.

**Beispiel:** Multiplikation zweier  $n$ -Bit Zahlen

Der Aufwand für das Finden der optimalen Reihenfolge ist NP-vollständig.

# Operationen auf BDDs (1)

Die Verknüpfung zweier BDDs basiert auf einer Erweiterung des Entwicklungssatzes:

$$h = f \circ g = x_i (f(x_i = 1) \circ g(x_i = 1)) \vee \overline{x_i} (f(x_i = 0) \circ g(x_i = 0))$$

Das Symbol  $\circ$  steht stellvertretend für einen beliebigen Booleschen Operator. ( $\wedge$ ,  $\vee$ ,  $\oplus$ ,  $\Leftrightarrow$ , usw.)

Die Beziehung wird auf die beiden BDDs für  $f$  und  $g$  startend mit den jeweiligen Wurzelknoten in der Variablenreihenfolge  $\tau$  angewendet.

## Bemerkung:

- Die Variablenreihenfolge muß für beide BDDs einheitlich sein.
- Die Verknüpfung von  $f$  und  $g$  zu  $h = f \circ g$  hat im worst case eine Bearbeitungsdauer, die dem Produkt der Größe der beiden Graphen entspricht ( $O(|f| \cdot |g|)$ ).

# Operationen auf BDDs (2)

Es seien  $v_f$  und  $v_g$  die zu einem bestimmten Zeitpunkt betrachteten Knoten der beiden Graphen:

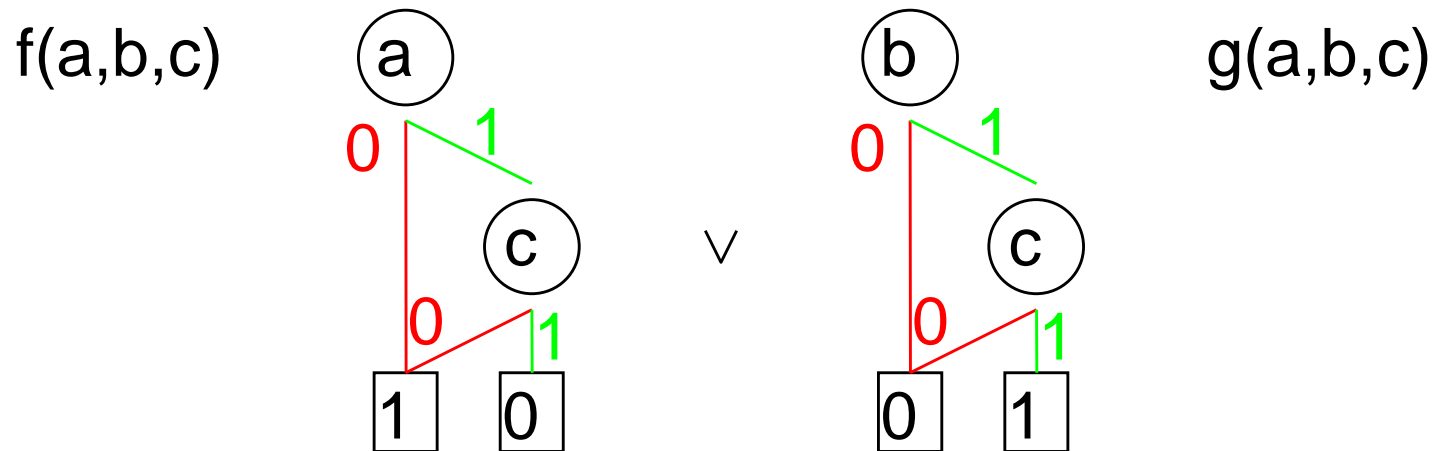
1. Sind beide Knoten Nonterminalknoten mit **gleichen Bezeichnern**, wird im Verknüpfungsgraphen der Knoten  $v = v_f = v_g$  erzeugt und der Algorithmus rekursiv auf die Kinder von  $v_f$  und  $v_g$  angewendet. Die Verknüpfung von  $n0(v_f)$  und  $n0(v_g)$  ergibt  $n0(v)$ , die Verknüpfung von  $n1(v_f)$  und  $n1(v_g)$  ergibt  $n1(v)$ . (*Entwicklungssatz vollständig angewendet*)
2. Sind beide Knoten Nonterminalknoten mit **verschiedenen Bezeichnern** oder ist nur einer der Knoten ein Nonterminalknoten, so sei  $v_f$  der Knoten der in der Variablenreihenfolge vor dem Knoten  $v_g$  kommt. Im Verknüpfungsgraphen wird der Knoten  $v = v_f$  erzeugt und der Algorithmus wird rekursiv auf die Kinder von  $v_f$  und  $v_g$  angewendet. Die Verknüpfung von  $n0(v_f)$  und  $v_g$  ergibt  $n0(v)$ , die Verknüpfung von  $n1(v_f)$  und  $v_g$  ergibt  $n1(v)$ . (**Entwicklungssatz auf f angewendet**)
3. Sind beide Knoten **Terminalknoten**, ergibt sich der Funktionswert aus der Verknüpfung beider Einzelfunktionen,  $W(v_f \circ v_g) = W(v_f) \circ W(v_g)$ .

# Beispiel – Operationen auf BDDs (1)

Die Verknüpfung

$$h(a, b, c) = \underbrace{(\bar{a} \vee \bar{c})}_{f(a,b,c)} \vee \underbrace{(b \wedge c)}_{g(a,b,c)}$$

wird bei einer angenommenen Variablenordnung  $\tau = a < b < c$  durch die folgenden ROBDDs dargestellt:



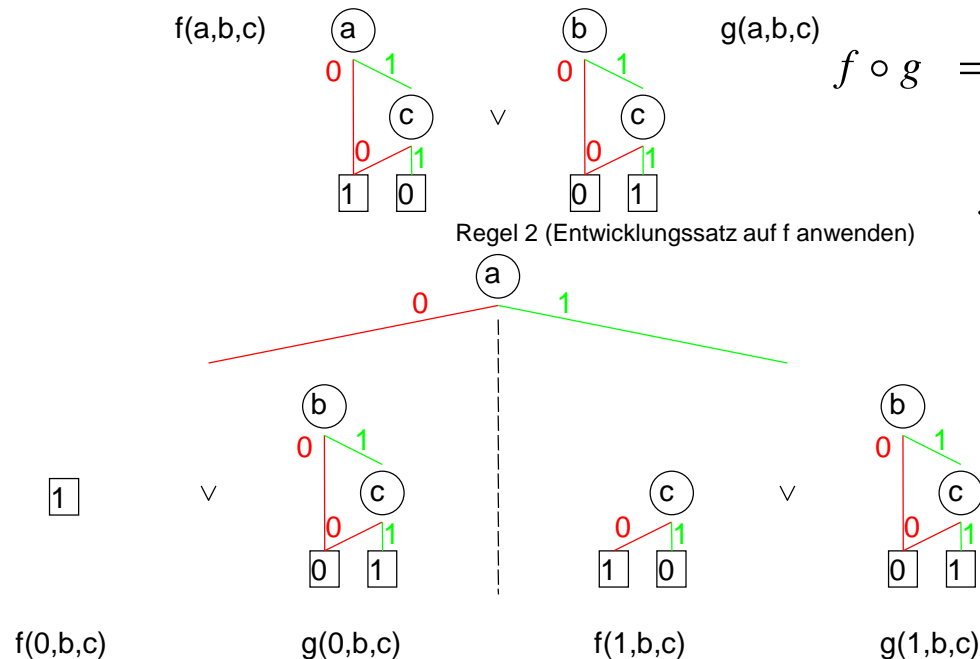
Entwicklungssatz:

$$h = f \circ g = x_i (f(x_i = 1) \circ g(x_i = 1)) \vee \bar{x}_i (f(x_i = 0) \circ g(x_i = 0))$$

# Beispiel – Operationen auf BDDs (2)

## Anwendung der Regel 2:

Die beiden Wurzelknoten der Funktionen  $f(a,b,c)$  und  $g(a,b,c)$  besitzen verschiedene Bezeichner und der Knoten  $a$  kommt in der Variablenreihenfolge vor dem Knoten  $b$ . Für beide Belegungen der Variablen  $a$  werden die Co-Faktoren der Funktionen  $f(a,b,c)$  und  $g(a,b,c)$  gebildet, indem die Kanten des Knoten  $a$  verfolgt werden.



$$f \circ g = x_i (f(x_i = 1) \circ g(x_i = 1)) \vee \overline{x_i} (f(x_i = 0) \circ g(x_i = 0))$$

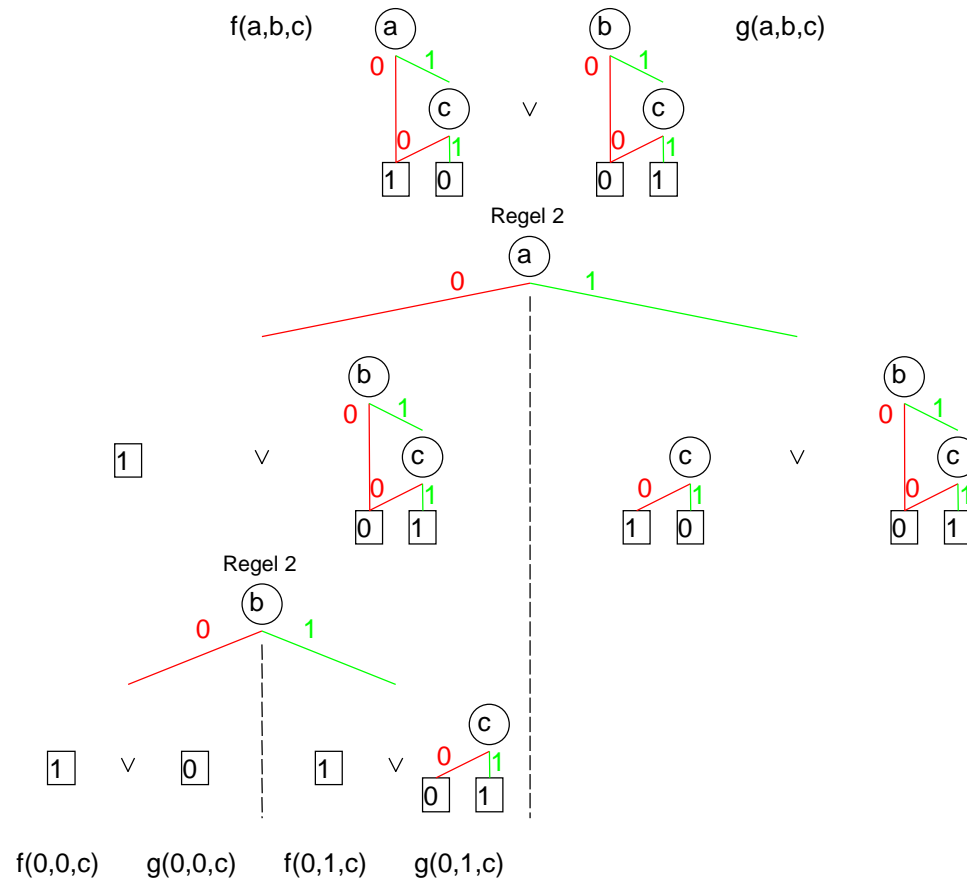
$$f \circ g = x_i (f(x_i = 1) \circ g) \vee \overline{x_i} (f(x_i = 0) \circ g)$$



# Beispiel – Operationen auf BDDs (3)

## Anwendung der Regel 2:

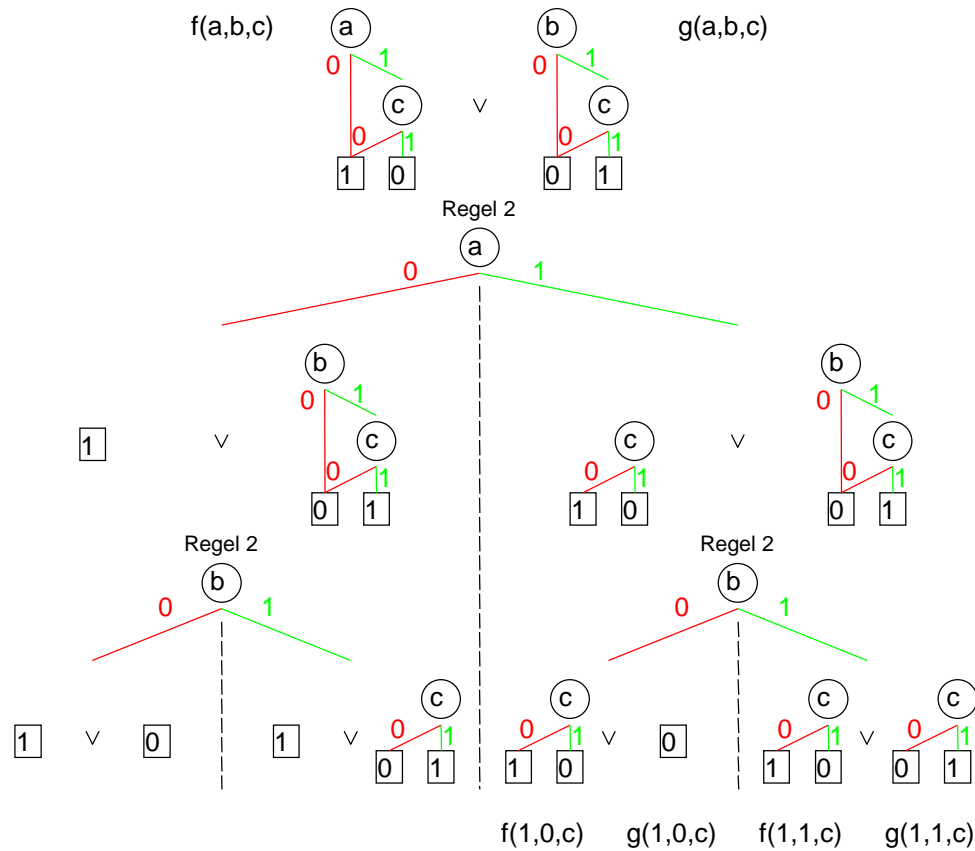
Die Co-Faktoren  $f(0,b,c)$  und  $g(0,b,c)$  besitzen verschiedene Bezeichner.



# Beispiel – Operationen auf BDDs (4)

## Anwendung der Regel 2:

Die Co-Faktoren  $f(1,b,c)$  und  $g(1,b,c)$  besitzen verschiedene Bezeichner.



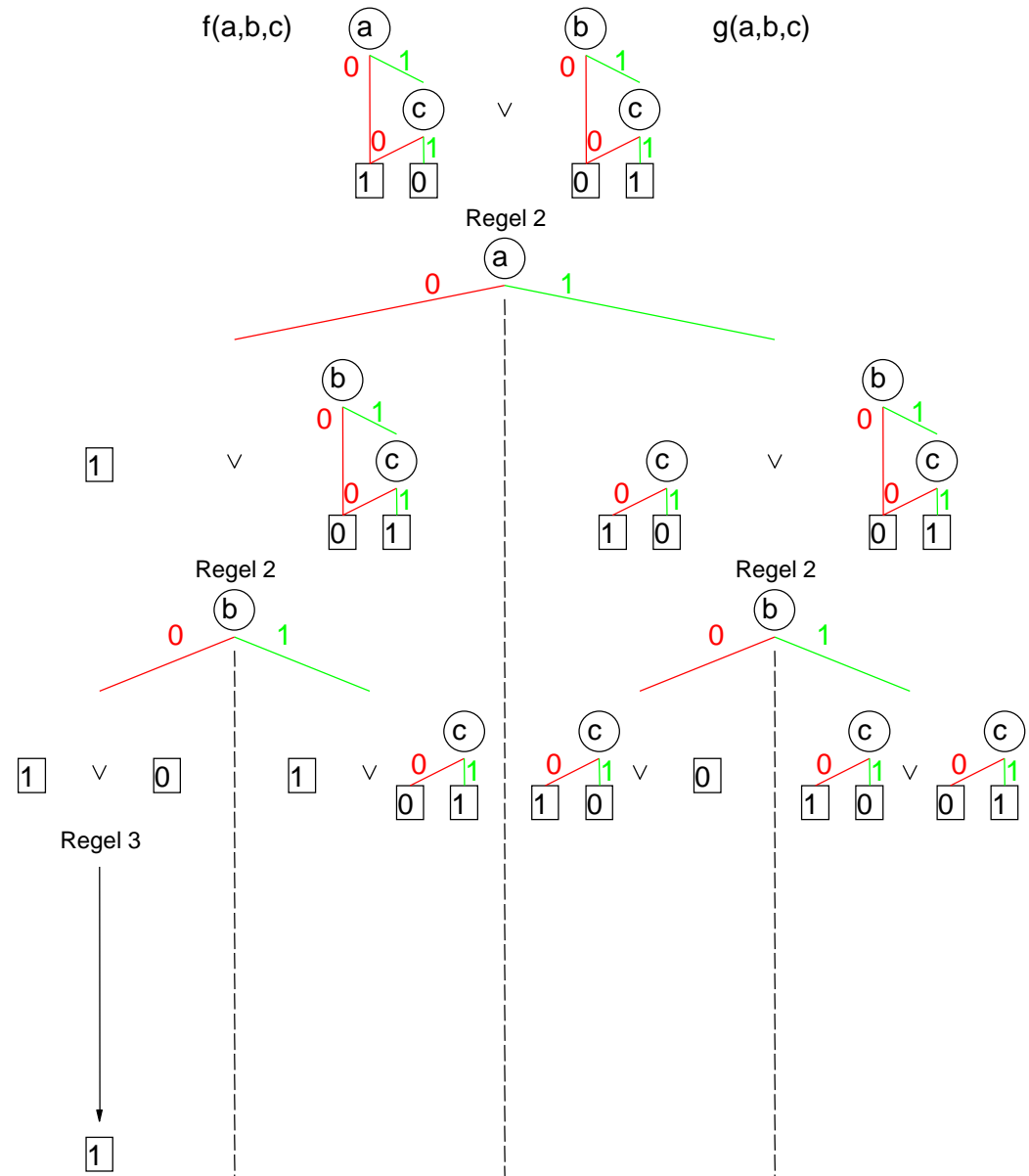
# Beispiel – Operationen auf BDDs (5)

## Anwendung der Regel 3:

Da es sich bei beiden Knoten um Terminalknoten handelt, ergibt sich der Wert der verknüpften Funktionen als Verknüpfung der Co-Faktoren  $f(0,0,c)$  und  $g(0,0,c)$ .

## Bemerkung:

Erst bei Anwendung der Regel 3 wird die Verknüpfung (in diesem Beispiel  $\vee$ ) durchgeführt!



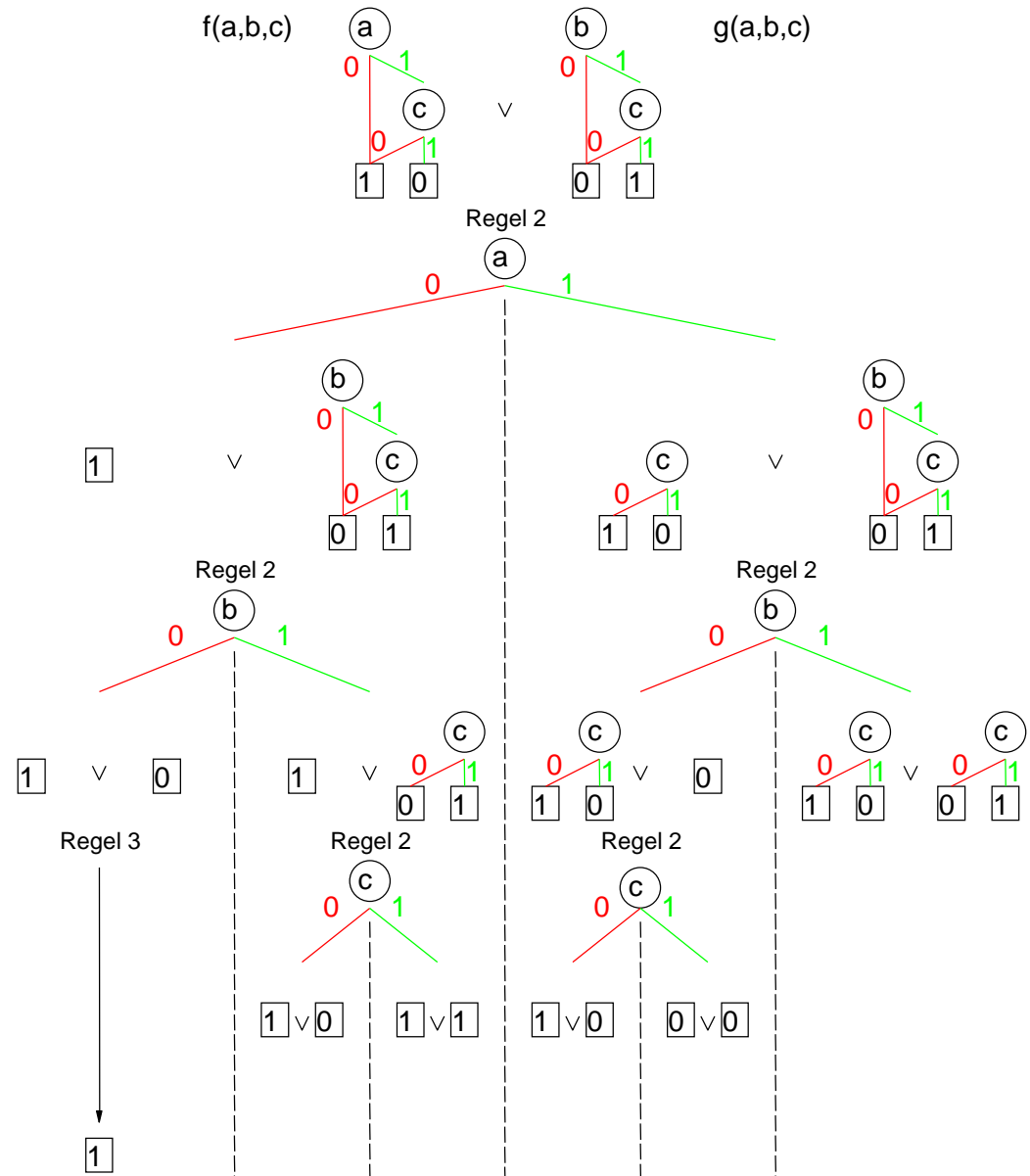
# Beispiel – Operationen auf BDDs (6)

## Anwendung der Regel 2:

Die Co-Faktoren  $f(0,1,c)$  und  $g(0,1,c)$  besitzen jeweils verschiedene Bezeichner.

## Anwendung der Regel 2:

Die Co-Faktoren  $f(1,0,c)$  and  $g(1,0,c)$  besitzen jeweils verschiedene Bezeichner.

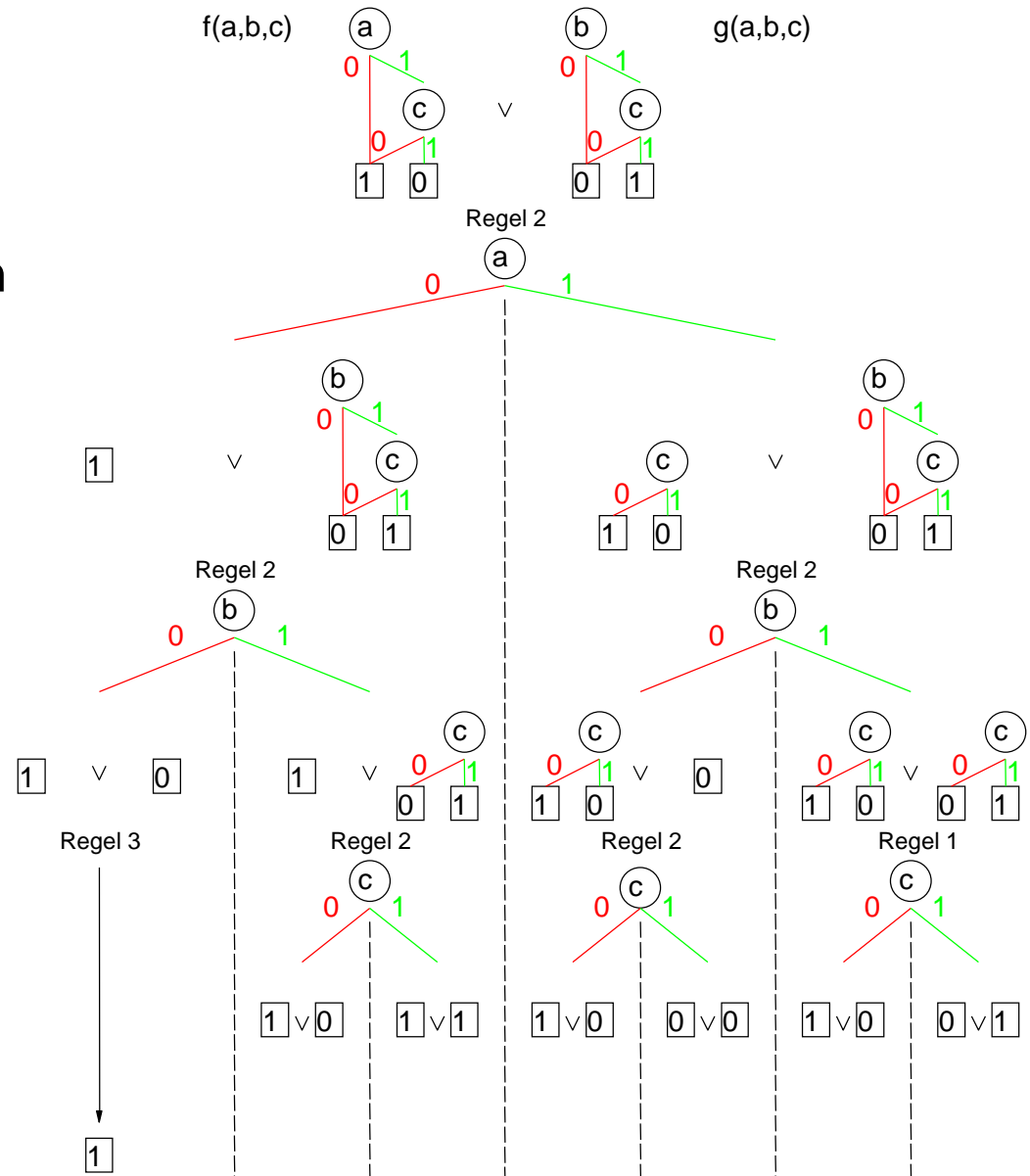


# Beispiel – Operationen auf BDDs (7)

## Anwendung der Regel 1:

Die beiden betrachteten Knoten besitzen den gleichen Bezeichner. Für beide Belegungen der Variablen  $c$  werden die Co-Faktoren bestimmt, indem die Kanten des Knoten  $c$  zu den Terminalen verfolgt werden.

$$f \circ g = x_i (f(x_i = 1) \circ g(x_i = 1)) \vee \overline{x_i} (f(x_i = 0) \circ g(x_i = 0))$$



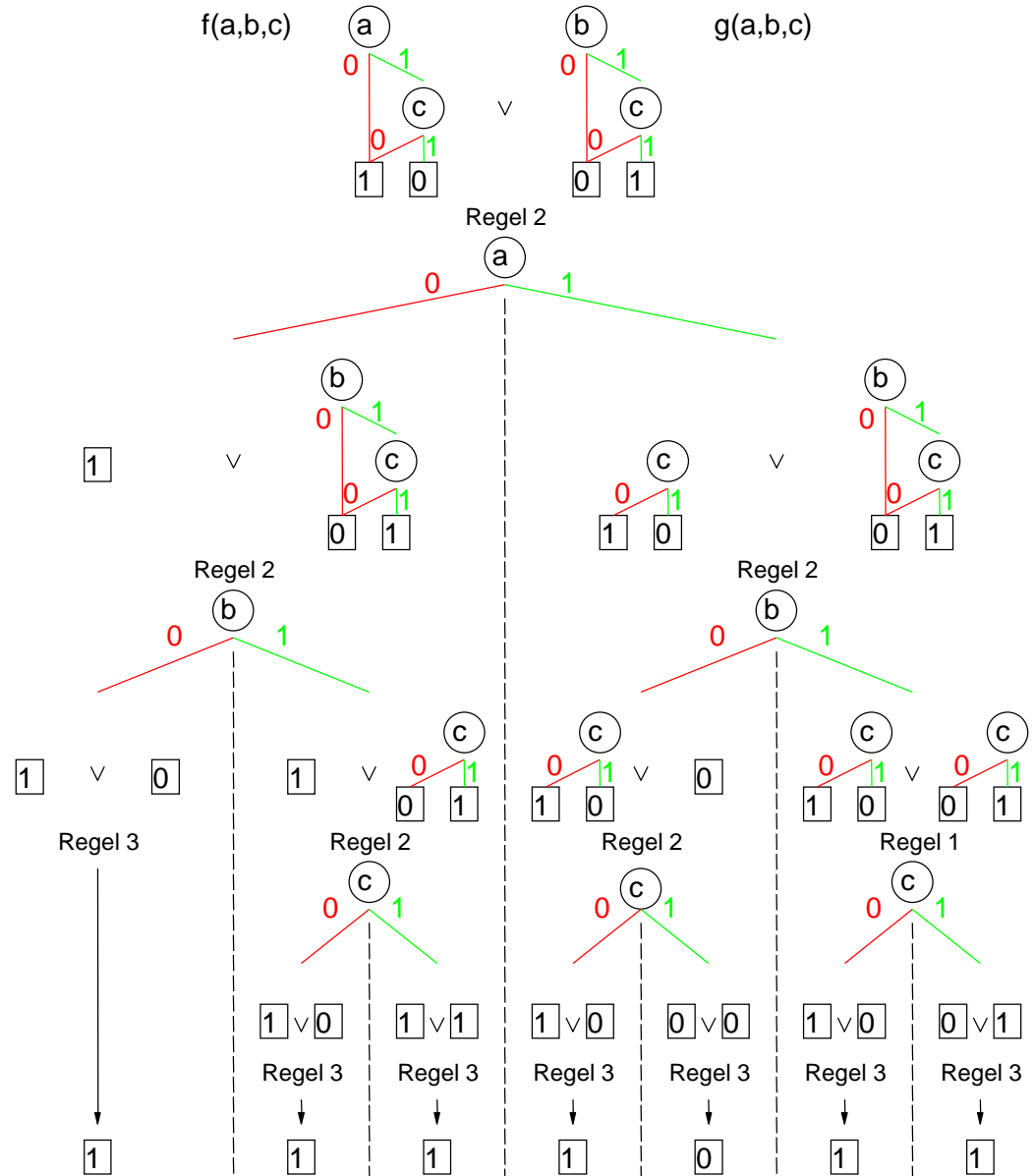
## Beispiel – Operationen auf BDDs (8)

## Anwendung der Regel 3:

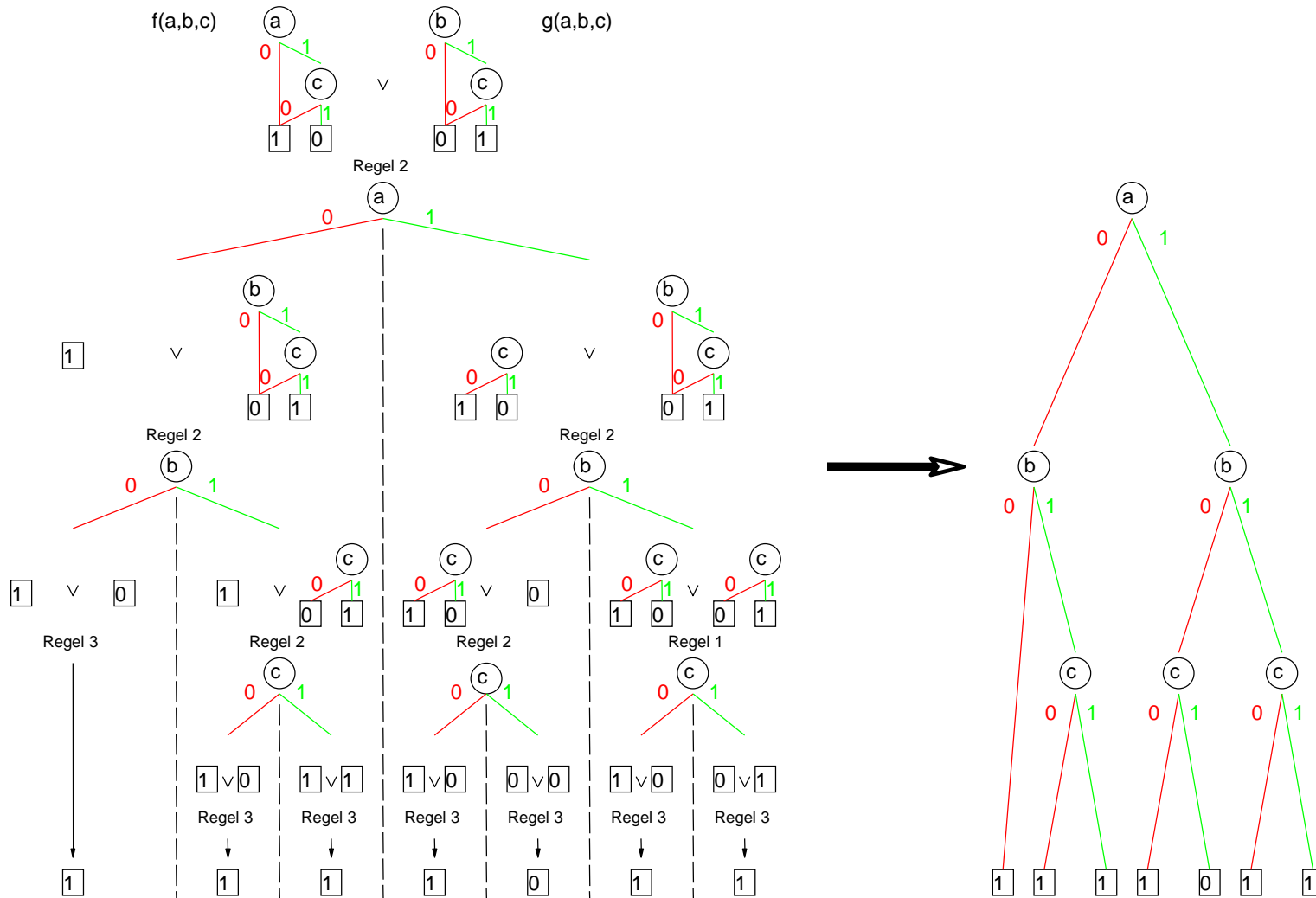
Bei allen noch nicht betrachteten Knoten handelt es sich um Terminale.

## Bemerkung:

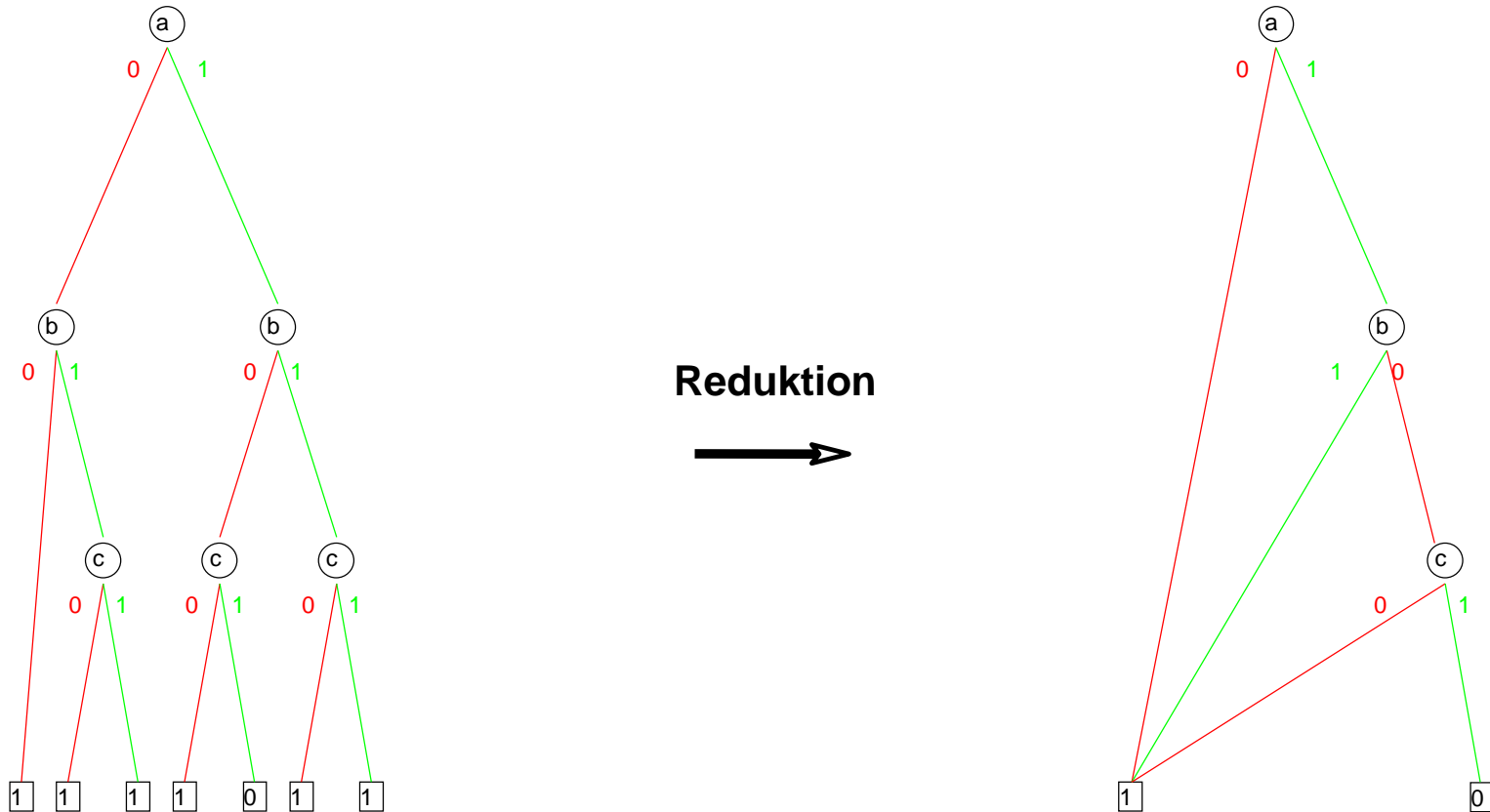
Der verknüpfte OBDD ist durch die erzeugte Knoten und Terminale gegeben.



# Beispiel – Operationen auf BDDs (9)



# Beispiel – Operationen auf BDDs (10)





## 3.4 Optimierung Boolescher Funktionen

Die Optimierung boolescher Funktionen kann nach folgenden Kriterien erfolgen:

- **Verzögerungszeit**

Die Zeit die ein Signal benötigt um vom Eingang durch alle Gatter hindurch zum Ausgang zu gelangen. Dies wird in späteren Kapiteln noch explizit behandelt. Ein grobes Maß ist die Verschachtelungstiefe der booleschen Ausdrücke.

- **Flächenbedarf**

Die Anzahl und Größe der Gatter. Ein Maß dafür wird in diesem Kapitel vorgestellt.

Die Verfahren zur Flächenoptimierung bei schon minimalem Zeitbedarf sind:

2.4.1 Graphisches Verfahren im KV-Diagramm

2.4.2 Verfahren von Quine/McCluskey

# Optimierung von Normalformen

Kanonische Normalformen lassen sich oft zu einfacheren, nicht kanonischen Normalformen umformen, die zum Teil oder ganz aus unvollständigen Termen bestehen. Dies gilt auch für nicht kanonische Normalformen.

**Beispiel:**

$$\begin{aligned} f(a, b, c, d) &= \bar{a}\bar{b}c\bar{d} \vee \bar{a}\bar{b}cd \vee \bar{a}b\bar{c}\bar{d} \vee \bar{a}bcd \vee abcd \\ &= \bar{a}\bar{b}c \vee \bar{a}bc \vee bcd \\ &= \bar{a}c \vee bcd \end{aligned}$$

Allerdings lässt sich nicht jede kanonische Normalform vereinfachen.

**Beispiel:**

$$g(a, b, c) = \bar{a}\bar{b}\bar{c} \vee \bar{a}b\bar{c} \vee abc \vee \bar{a}\bar{b}c$$

# Optimierung von Normalformen (2)

## Satz von Quine:

Eine minimale Normalform besteht aus einer Teilmenge aller Primimplikanten.

## Vorgehensweise zur Optimierung von Normalformen:

1. Alle Primimplikanten bestimmen.
2. Auswahl einer kostenminimalen Teilmenge aller Primimplikanten bestimmen, die alle Minterme der Funktion überdeckt.

## Vergleichskriterium (Kosten): $\Lambda = \# \text{Literale} + \# \text{Terme}$

#Literale = Anzahl der Eingänge der ersten Ebene einer zweistufigen Implementierung

#Terme = Anzahl der Eingänge der zweiten Ebene einer zweistufigen Implementierung

Ziel der Optimierung von Normalformen ist die Minimierung von  $\Lambda$ .

# Beispiel – Vergleichskriterium

Für eine Implementierung der Funktion

$$f(a, b, c, d) = \bar{a}c \vee bcd$$

werden 1 AND-Gatter mit drei Eingängen, 1 AND-Gatter mit zwei Eingängen und 1 OR-Gatter mit zwei Eingängen benötigt.

$$\Lambda = \underbrace{2+3}_{\text{\#Literale}} + \underbrace{2}_{\text{\#Terme}} = 7$$

## 3.4.1 Graphisches Verfahren

Bei der graphischen Methode zur Minimierung von Schaltnetzen ist vor allem die Intuition des Schaltungsentwicklers gefragt, da es auf das Erkennen von Symmetrien ankommt.

Diese Vorgehensweise ist nur für Schaltungen geringer Komplexität geeignet.

### Vorgehensweise:

1. Ermittlung aller Primimplikanten  $P(f)$  einer Funktion  $f$
2. Auswahl derjenigen  $\{p_1, p_2, \dots, p_k\} \in P(f)$ , für die
  - (a)  $NF(f) = p_1 \vee p_2 \vee \dots \vee p_k = f$
  - (b)  $\Lambda = \min$

Die Menge  $P(f)$  lässt sich direkt aus dem KV-Diagramm entnehmen. Die Suche nach  $NF_{\min}(f)$  lässt sich durch Klassifizierung der Primimplikanten erleichtern.

# Klassifizierung der Primimplikanten

## Kern-Primimplikant $P_K$ :

Ein Primimplikant  $p$  ist ein Kern-Primimplikant, d.h.  $p \in P_K$ , falls  $p$  von der Disjunktion aller übrigen  $p \in P(f)$  nicht überdeckt wird.

**Bemerkung:** Ein Kern-Primimplikant überdeckt mindestens einen Minterm, der von keinem anderen Primimplikanten überdeckt wird.

## Absolut eliminierbare Primimplikanten $P_A$ :

Ein Primimplikant  $p$  ist ein absolut eliminierbarer Primimplikant, d.h.  $p \in P_A$ , falls  $p$  von der Disjunktion aller  $P_K$  überdeckt wird.

**Bemerkung:** Ein absolut eliminierbarer Primimplikant wird vollständig von Kern-Primimplikanten überdeckt.

## Relativ eliminierbare Primimplikanten $P_R$ :

Ein Primimplikant  $p$  ist ein relativ eliminierbarer Primimplikant, d.h.  $p \in P_R$ , falls  $p$  von der Disjunktion von  $P(f) - P_A$  überdeckt wird.

**Bemerkung:** Ein relativ eliminierbarer Primimplikant überdeckt nur mehrfach überdeckte Minterme, wobei nicht alle von Kern-Primimplikanten überdeckt werden.

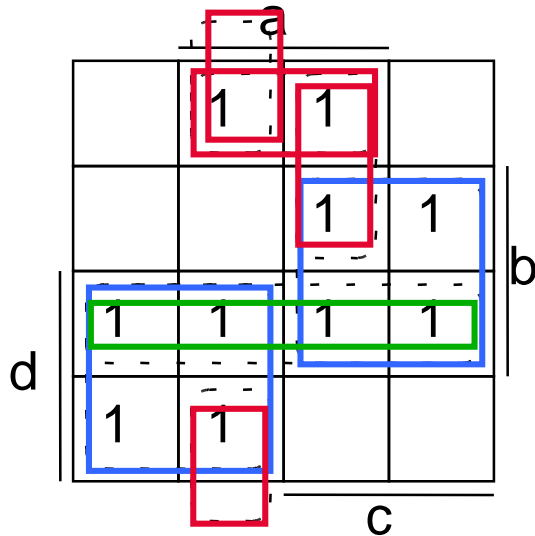
# Minimale Normalform

## Satz:

Die minimale Normalform  $NF_{min}(f)$  besteht nur noch aus den Kern-Primimplikanten  $P_K$  und einer Auswahl relativ eliminierbarer Primimplikanten  $p \in P_R$  derart, dass kein Primimplikant von der Disjunktion der übrigen  $p \in NF_{min}(f)$  überdeckt wird.

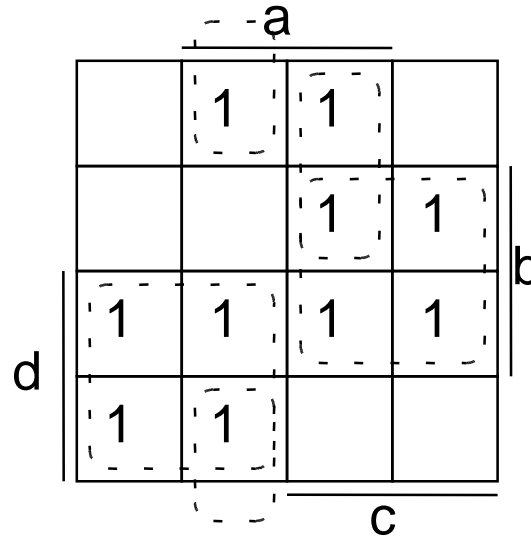
Im allgemeinen erfüllen für eine gegebene Funktion  $f$  mehrere  $NF$  diese Bedingung. Von ihnen wird diejenige ausgewählt, für die der Kostenwert  $\Delta$  minimal ist.

# Beispiel – Minimale Normalform

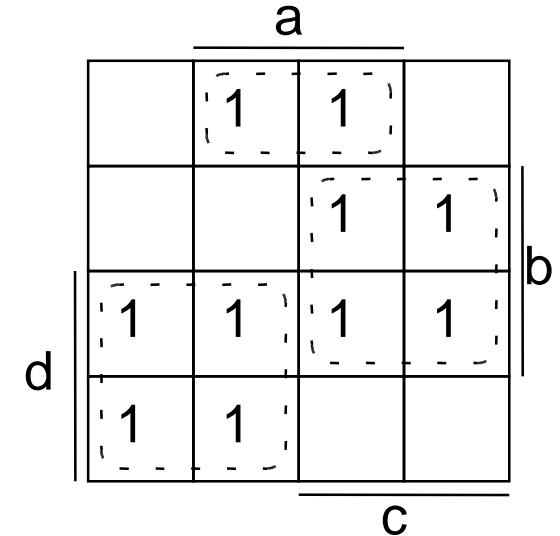


Primimplikanten

$$P = \{b c, \bar{c} d, a \bar{b} \bar{c}, a c \bar{d}, a \bar{b} \bar{d}, b d\}$$



irredundante NF



irredundante und  
minimale NF

$$P_K = \{b c, \bar{c} d\}$$

$$P_A = \{b d\}$$

$$P_R = \{a \bar{b} \bar{c}, a c \bar{d}, a \bar{b} \bar{d}\}$$

$$NF_{min}(f) = b c \vee \bar{c} d \vee a \bar{b} \bar{d}$$

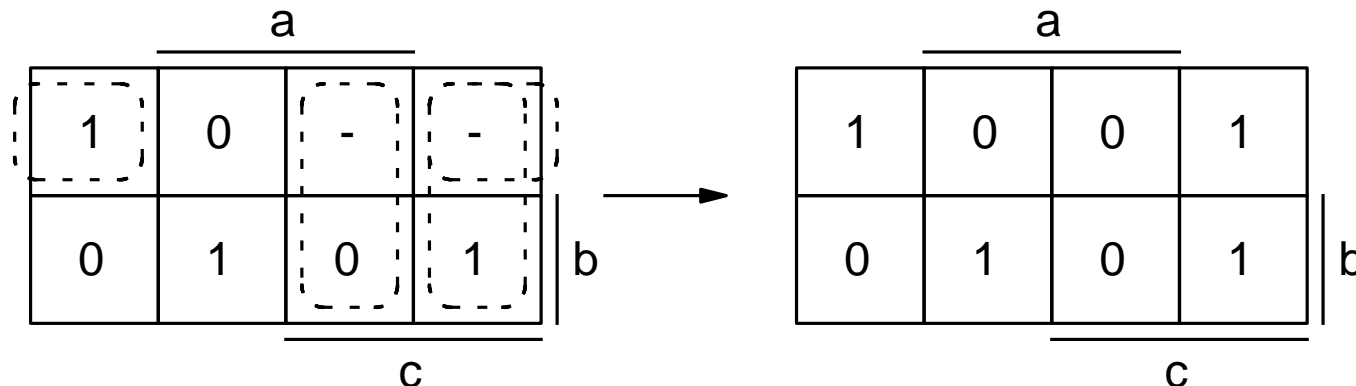


# Optimierung unvollst. spezif. Funktionen

## Vorgehensweise:

1. Alle irredundanten disjunktiven Normalformen für die Funktion aufstellen.
2. Für jede Form nach 1 diejenigen don't cares zu Implikanten machen, die zur Elimination von Literalen in einem Term oder zur Verschmelzung mehrerer Terme führen.
3. Auswahl einer minimalen Form aus 2.

## Beispiel:



# Ermittlung minimaler konjunktiver NF

## Vorgehensweise:

1. Inversion der gegebenen Funktion  $f$  (vertauschen von 0 und 1).
2. Ermittlung der minimalen disjunktiven  $NF$  für  $f$  nach bekanntem Verfahren.
3. Algebraische Umwandlung der negierten disjunktiven Minimalform in eine konjunktive  $NF$  mit Hilfe des Gesetzes von DeMorgan oder Anwendung des Shannon'schen Inversionssatzes.

$$\overline{f(a,b,c)}$$

1	0	0	1
1	1	0	1

a

b

c

$$\overline{f(a,b,c)}$$

0	1	1	0
0	0	1	0

a

b

c

$$\overline{f} = a\overline{b} \vee ac$$

$$f = (\overline{a} \vee b) \wedge (\overline{a} \vee \overline{c})$$

## 3.4.2 Verfahren von Quine/McCluskey

### Vorgehensweise:

1. Ermittlung aller Primimplikanten  $P(f)$  einer Funktion  $f$
2. Auswahl derjenigen  $\{p_1, p_2, \dots, p_k\} \in P(f)$ , für die
  - a)  $NF(f) = p_1 \vee p_2 \vee \dots \vee p_k = f$
  - b)  $\Lambda = \min$

# Bestimmung aller Primimplikanten

## Idee von Quine:

Wiederholte Anwendung aller möglichen Verschmelzungsbeziehungen:

$$M_i \vee M_j = M_{i,j}(x_k \vee \overline{x_k}) = M_{i,j}$$

## Hilfreiche Verfeinerung von McCluskey:

1. Statt Minterme werden die eindeutig zugeordneten Argument- $n$ -Tupel notiert.
2. Terme ( $l$ -Tupel) gleicher Länge  $l$  kommen jeweils in eine eigene Spalte. Durch Verschmelzung weggefallene Variablen werden durch „-“ gekennzeichnet.

# Algorithmus

## # Zunächst alle Primimplikanten bestimmen

- 1) Aufstellen der Mintermform
- 2) Bestimmung der Primimplikanten durch Verschmelzung

## # Ab hier nun die Lösung des Überdeckungsproblems

**Solange** noch nicht alle Minterme abgedeckt

**Solange** noch Veränderungen in der Primimplikanten-Minterm-Tabelle

- 3) Aufstellen der Primimplikanten-Minterm-Tabelle
- 4) Aufsuchen der Kern-Primimplikanten
- 5) Reduzierte Überdeckungsmatrix
- 6) Auslassen doppelt erfasster Minterme
- 7) Auslassen bereits erfasster Terme
- 8) Auslassen ungünstiger Primimplikanten
- 9) Primimplikanten zweiter Art

**Ende** von „Solange noch Veränderungen

- 11) Aufsuchen der einfachsten Primimplikantenkombination

**Ende** Solange noch nicht alle Minterme abgedeckt

# Bestimmung aller Primiplikanten

## 1) Aufstellen der Mintermform

Eine gegebene Boolesche Funktion wird durch Anwendung der inversen Elemente in die Mintermform überführt.

$$\begin{aligned} f &= \textcircled{abc} \vee \bar{a}cd \vee bd \vee ac\bar{d} \vee \bar{a}\bar{b}\bar{c}\bar{d} \vee \bar{b}c\bar{d} \\ &= \bar{a}\bar{b}\bar{c}\bar{d} \vee \bar{a}\bar{b}c\bar{d} \vee \bar{a}b\bar{c}d \vee \bar{a}b\bar{c}\bar{d} \vee \bar{a}bcd \vee \\ &\quad \textcircled{ab\bar{c}\bar{d}} \vee abcd \vee \textcircled{ab\bar{c}d} \vee \bar{a}bcd \vee abcd \end{aligned}$$

# Bestimmung aller Primimplikanten

## 2) Bestimmung der Primimplikanten durch Verschmelzung

- Die Minterme werden, sortiert nach der Anzahl der Einsen, in einer Liste (Implikantentabelle) untereinander angeordnet.
- Danach werden alle Minterme dahingehend verglichen, ob sie sich nur in einer Variable unterscheiden, die einmal negiert und einmal nicht negiert vorkommt.

Hierfür müssen nur Minterme benachbarter Gruppen verglichen werden.

$\bar{a}\bar{b}\bar{c}\bar{d}$	(0000)
$\bar{a}\bar{b}c\bar{d}$	(0010)
$\bar{a}\bar{b}cd$	(0011)
$\bar{a}b\bar{c}d$	(0101)
$a\bar{b}c\bar{d}$	(1010)
$ab\bar{c}\bar{d}$	(1100)
$abc\bar{d}$	(1110)
$ab\bar{c}d$	(1101)
$\bar{a}bcd$	(0111)
$abcd$	(1111)

# Bestimmung aller Primimplikanten

- Existieren solche Terme, dann werden sie zusammengefasst, wobei ein verkürzter Term entsteht. Die verkürzten Terme werden in einer zweiten Spalte angeordnet.
- Die Terme  $\bar{a}\bar{b}\bar{c}\bar{d}$  und  $\bar{a}\bar{b}c\bar{d}$  unterscheiden sich nur in der Variablen  $c$  und können daher durch den verkürzten Term  $\bar{a}\bar{b}\bar{d}$  dargestellt werden.
- Die in die Zusammenfassung eingegangenen Terme werden durch einen Stern markiert.

$\bar{a}\bar{b}\bar{c}\bar{d}$	(0000) *	$\bar{a}\bar{b}\bar{d}$	(00-0)
$\bar{a}\bar{b}c\bar{d}$	(0010) *	$\bar{a}\bar{b}c$	(001-)
		$\bar{b}c\bar{d}$	(-010)
$\bar{a}\bar{b}cd$	(0011) *	$\bar{a}bd$	(01-1)
$\bar{a}b\bar{c}\bar{d}$	(0101) *	$\bar{a}cd$	(0-11)
$a\bar{b}c\bar{d}$	(1010) *	$b\bar{c}\bar{d}$	(-101)
$ab\bar{c}\bar{d}$	(1100) *	$ac\bar{d}$	(1-10)
		$ab\bar{d}$	(11-0)
		$ab\bar{c}$	(110-)
$abc\bar{d}$	(1110) *	$abd$	(11-1)
$ab\bar{c}d$	(1101) *	$bcd$	(-111)
$\bar{a}bcd$	(0111) *	$abc$	(111-)
$abcd$	(1111) *		



# Bestimmung aller Primimplikanten

- Anschließend werden die verkürzten Terme verkürzt. Das Verfahren terminiert, wenn keine Verkürzung mehr möglich ist.

$\bar{a}\bar{b}\bar{c}\bar{d}$	(0000) *	$\bar{a}\bar{b}\bar{d}$	(00-0)	
$\bar{a}\bar{b}c\bar{d}$	(0010) *	$\bar{a}\bar{b}c$	(001-)	
		$\bar{b}c\bar{d}$	(-010)	
$\bar{a}\bar{b}cd$	(0011) *	$\bar{a}bd$	(01-1) *	$bd$ (-1-1)
$\bar{a}b\bar{c}d$	(0101) *	$\bar{a}cd$	(0-11)	$ab$ (11--)
$a\bar{b}c\bar{d}$	(1010) *	$b\bar{c}d$	(-101) *	
$ab\bar{c}\bar{d}$	(1100) *	$acd$	(1-10)	
		$ab\bar{d}$	(11-0) *	
		$ab\bar{c}$	(110-) *	
$abc\bar{d}$	(1110) *	$abd$	(11-1) *	
$ab\bar{c}d$	(1101) *	$bcd$	(-111) *	
$\bar{a}bcd$	(0111) *	$abc$	(111-) *	
$abcd$	(1111) *			

# Bestimmung aller Primimplikanten

- Abschließend werden alle Terme mit einem Stern markiert, die zu mindestens einer Verkürzung herangezogen wurden.
- Alle nicht markierten Terme werden als Primimplikanten oder Primterme bezeichnet.

$\bar{a}\bar{b}\bar{c}\bar{d}$	(0000)	*	$\bar{a}\bar{b}\bar{d}$	(00-0)	
$\bar{a}\bar{b}c\bar{d}$	(0010)	*	$\bar{a}\bar{b}c$	(001-)	
			$\bar{b}c\bar{d}$	(-010)	
$\bar{a}\bar{b}cd$	(0011)	*	$\bar{a}bd$	(01-1)	*
$\bar{a}b\bar{c}d$	(0101)	*	$\bar{a}cd$	(0-11)	
$a\bar{b}c\bar{d}$	(1010)	*	$b\bar{c}d$	(-101)	*
$ab\bar{c}\bar{d}$	(1100)	*	$acd$	(1-10)	
			$ab\bar{d}$	(11-0)	*
			$ab\bar{c}$	(110-)	*
$abc\bar{d}$	(1110)	*	$abd$	(11-1)	*
$ab\bar{c}d$	(1101)	*	$bcd$	(-111)	*
$\bar{a}bcd$	(0111)	*	$abc$	(111-)	*
$abcd$	(1111)	*			

# Überdeckungsproblem

Aus der Menge der Primimplikanten muß die Teilmenge bestimmt werden, deren Primimplikanten alle Minterme überdeckt und zu minimalen Kosten führt. Dieses Problem kann mit dem Rucksackproblem verglichen werden und ist somit NP-hart.

## Heuristik:

Aus der Menge der Primimplikanten sukzessive solche auswählen, die möglichst viele, noch nicht überdeckte Minterme neu überdecken. Für die Lösung der Heuristik gilt:

$$1 \leq \frac{\# \text{ der durch Heuristik gewählten } P}{\# \text{ der minimal erforderlichen } P}$$

# Überdeckungsproblem

## 3) Aufstellen der Primimplikanten-Minterm-Tabelle

Primimplikanten und Minterme werden in einer Tabelle derart angeordnet, dass die Minterme die Spaltenbezeichnung bilden und die Primimplikanten die Zeilenbezeichnung.

	$\bar{a}\bar{b}\bar{c}\bar{d}$	$\bar{a}\bar{b}c\bar{d}$	$\bar{a}b\bar{c}\bar{d}$	$\bar{a}b\bar{c}d$	$\bar{a}bc\bar{d}$	$\bar{a}bcd$	$ab\bar{c}\bar{d}$	$ab\bar{c}d$	$abc\bar{d}$	$abcd$
$\bar{a}\bar{b}\bar{d}$										
$\bar{a}\bar{b}c$										
$\bar{b}c\bar{d}$										
$\bar{a}c\bar{d}$										
$ac\bar{d}$										
$bd$										
$ab$										

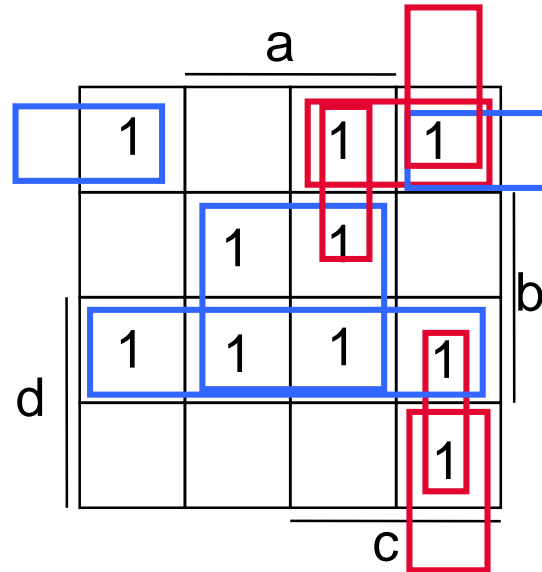
# Überdeckungsproblem

Jedes Feld der Tabelle, in dem sich die Zeile eines Primimplikanten mit der Spalte eines in ihm enthaltenen Minterms kreuzt, wird mit einem Stern gekennzeichnet.

	$\bar{a}\bar{b}\bar{c}\bar{d}$	$\bar{a}\bar{b}c\bar{d}$	$\bar{a}b\bar{c}\bar{d}$	$\bar{a}b\bar{c}d$	$\bar{a}b\bar{c}\bar{d}$	$a\bar{b}\bar{c}\bar{d}$	$a\bar{b}c\bar{d}$	$a\bar{b}c\bar{d}$	$a\bar{b}cd$	$ab\bar{c}\bar{d}$
$\bar{a}\bar{b}\bar{d}$	*	*								
$\bar{a}\bar{b}c$		*	*							
$\bar{b}c\bar{d}$		*			*					
$\bar{a}c\bar{d}$			*						*	
$a\bar{c}\bar{d}$					*		*			
$b\bar{d}$				*				*	*	*
$a\bar{b}$						*	*	*		*

Die Disjunktion einer Anzahl von Primimplikanten, deren Sterne das Schema horizontal überdecken ist äquivalent zur Ausgangsfunktion.  
Aufgabe: Finde die einfachste horizontale Überdeckung hinsichtlich  $\wedge$ .

# Zum Vergleich: Alle Primimplikanten im KV-Diagramm



Kern-Primimpl.

Relative eliminierbare P.

# Überdeckungsmatrix

## 4) Aufsuchen der Kern-Primimplikanten

Wenn in der Überdeckungsmatrix Spalten mit nur einem einzigen Stern auftreten, so bedeutet das, dass der Primimplikant der zugehörigen Zeile in der endgültigen Form enthalten sein muß. Bei den Primimplikanten handelt es sich um Kern-Primimplikanten. Im 4.Schritt werden alle Spalten ermittelt, die nur einen Stern enthalten und die zugehörigen Kern-Primimplikanten markiert.

**Kern-Primimplikanten:**

$$\bar{a}\bar{b}\bar{d}, \quad bd, \quad ab$$

# Überdeckungsmatrix

## 5) Reduzierte Überdeckungsmatrix

Im 5.Schritt wird aus der Überdeckungsmatrix eine reduzierte Überdeckungsmatrix gebildet, in der die Kern-Primimplikanten und *alle* überdeckten Minterme nicht mehr auftreten.

	$\bar{a}\bar{b}cd$	$a\bar{b}c\bar{d}$
$\bar{a}\bar{b}c$	*	
$\bar{b}c\bar{d}$		*
$\bar{a}cd$	*	
$ac\bar{d}$		*



# Überdeckungsmatrix

## 6) Auslassen doppelt erfasster Minterme

Hat in der Überdeckungsmatrix eine Spalte in mindestens den Feldern Sterne, in denen auch eine andere Spalte Sterne hat, dann kann dieser Minterm ausgelassen werden (**Spaltendominanz**).

**Beispiel:**

	$\bar{a}\bar{b}\bar{c}d$	$\bar{a}b\bar{c}\bar{d}$	$\bar{a}\bar{b}c\bar{d}$	$\bar{a}bc\bar{d}$
$\bar{a}\bar{b}\bar{d}$			*	
$\bar{a}b\bar{c}$				*
$\bar{a}d$	*			
$\bar{a}\bar{c}$	*	*	*	
$\bar{a}b$	*	*		*
	↑↑			

Die Spalte  $\bar{a}b\bar{c}d$  wird von der Spalte  $\bar{a}b\bar{c}\bar{d}$  dominiert und kann entfernt werden.

# Überdeckungsmatrix

## 7) Auslassen bereits erfasster Terme

Durch den 6.Schritt können Zeilen entstehen, die keinen Stern mehr enthalten. Die entsprechenden Primimplikanten können weggelassen werden.

# Überdeckungsmatrix

## 8) Auslassen ungünstiger Primimplikanten

Ein Primimplikant  $p_1$  dominiert einen Primimplikanten  $p_2$ , wenn mindestens alle Minterme, welche von  $p_2$  überdeckt werden auch von  $p_1$  überdeckt werden (**Zeilendominanz**). Primimplikanten, welche von Primimplikanten mit geringeren oder gleichen Kosten dominiert werden, können weggelassen werden.

**Beispiel:**

		$\bar{a}\bar{b}\bar{c}d$	$\bar{a}b\bar{c}\bar{d}$	$\bar{a}\bar{b}c\bar{d}$	$\bar{a}b\bar{c}d$
$\Rightarrow$	$\bar{a}\bar{b}\bar{d}$			*	
$\Rightarrow$	$\bar{a}b\bar{c}$				*
$\Rightarrow$	$\bar{a}d$	*			
	$\bar{a}\bar{c}$	*	*	*	
	$\bar{a}b$	*	*		*

Die Zeilen  $\bar{a}\bar{b}\bar{d}$ ,  $\bar{a}b\bar{c}$  und  $\bar{a}d$  können entfernt werden.

# Überdeckungsmatrix

## 8) Auslassen ungünstiger Primimplikanten

Ursprüngliches Beispiel

		$\bar{a}\bar{b}cd$	$a\bar{b}c\bar{d}$
	$\bar{a}\bar{b}c$	*	
	$\bar{b}c\bar{d}$		*
$\Rightarrow$	$\bar{a}cd$	*	
$\Rightarrow$	$acd$		*

# Überdeckungsmatrix

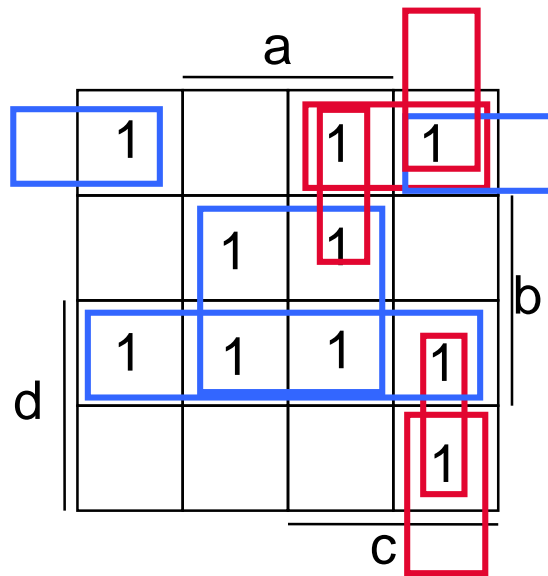
## 9) Primimplikanten zweiter Art

Nach der Durchführung von Schritt 8 können Spalten auftreten, die wiederum nur einen Stern enthalten. Die dazugehörigen Implikanten heißen Primimplikanten zweiter Art.

	$\bar{a}\bar{b}cd$	$a\bar{b}c\bar{d}$
$\bar{a}\bar{b}c$	*	
$\bar{b}c\bar{d}$		*

Minimale Endform:  $f = \bar{a}\bar{b}\bar{d} \vee bd \vee ab \vee \bar{a}\bar{b}c \vee \bar{b}c\bar{d}$

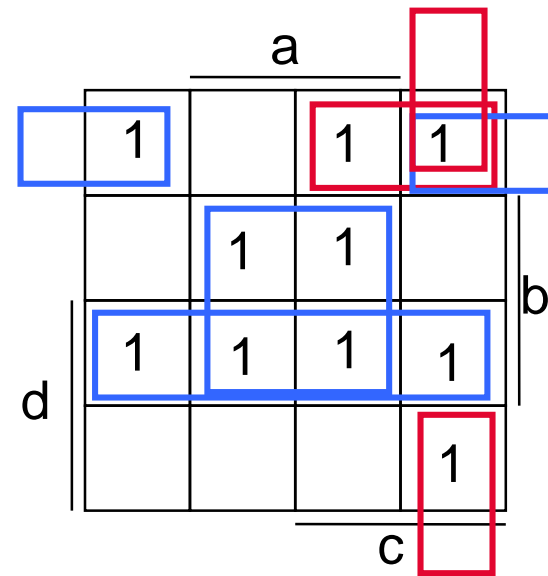
# Das ganze im KV-Diagramm



Primimplikanten

**Kern-Primimplikanten**

**Relative eliminierbare Pi.**



Minimale Überdeckung

## Bemerkung:

im KV-Diagramm können  
die Kosten nur indirekt  
abgelesen werden

# Überdeckungsmatrix

## 10) Iterative Überdeckung

Die Schritte 5-9 werden so lange durchgeführt, bis sie terminieren.

## 11) Aufsuchen der einfachsten Primimplikantenkombination

Finde die einfachste (im Sinne der Kosten) horizontale Überdeckung der Überdeckungsmatrix aus Schritt 10, die alle noch nicht erfassten Minterme überdeckt.

Schritt 11 ist der einzige unsystematische in diesem Verfahren.

# Überdeckungsmatrix

## 11) Aufsuchen der einfachsten Primimplikantenkombination

Anderes Beispiel mit 5 Variablen:

$$\begin{aligned} \text{Die Funktion } f = & bcd\bar{e} \vee bc\bar{d}e \vee b\bar{c}de \vee \bar{b}cde \vee \\ & b\bar{c}\bar{d}\bar{e} \vee \bar{b}c\bar{d}\bar{e} \vee \bar{b}\bar{c}d\bar{e} \vee \bar{b}\bar{c}\bar{d}e \vee \\ & \bar{a}b\bar{c} \vee \bar{a}\bar{b}c \end{aligned}$$

liefert nach allen Schritten bis einschließlich 10) folgende Tabelle:

		$\bar{a}\bar{b}c\bar{d}e$	$\bar{a}\bar{b}cd\bar{e}$	$\bar{a}b\bar{c}\bar{d}e$	$\bar{a}b\bar{c}d\bar{e}$
$\Rightarrow$	$\bar{a}b\bar{c}$			*	*
$\Rightarrow$	$\bar{a}\bar{b}c$	*	*		
	$\bar{a}d\bar{e}$		*		*
	$\bar{a}\bar{d}e$	*		*	