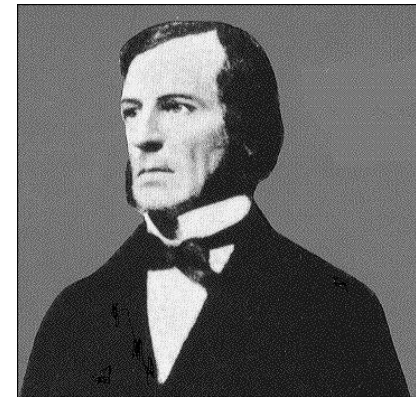


3. Boolesche Algebra

- **hier nur kurze Einführung**
 - Details siehe Vorlesung "Logik und diskrete Strukturen"
- **George Boole**
 - 1815 - 1864
 - britischer Mathematiker und Philosoph
 - begründete moderne mathematische Logik
- **Edward Vermilye Huntington**
 - 1874 - 1952
 - amerikanischer Mathematiker
 - konnte die Boolesche Algebra auf vier Axiome zurückführen



Boolesche Algebra

Gegeben: Menge V , Operatoren $\bullet, + : V \times V \rightarrow V$

V heißt Boolesche Algebra,
wenn die folgenden vier Huntingtonschen Axiome gelten:

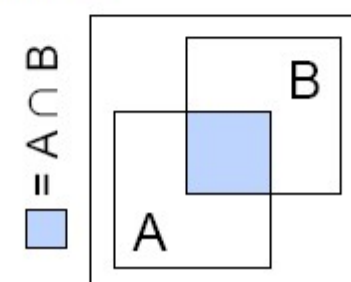
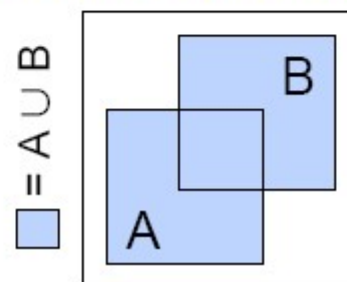
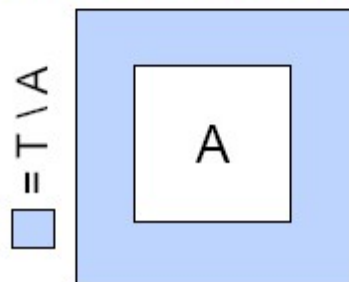
- Kommutativgesetze (K):
 $a \bullet b = b \bullet a$
 $a + b = b + a$
- Distributivgesetze (D):
 $a \bullet (b + c) = (a \bullet b) + (a \bullet c)$
 $a + (b \bullet c) = (a + b) \bullet (a + c)$
- Neutrale Elemente (N):
Es existieren $e, n \in V$ mit
 $a \bullet e = a$ und $a + n = a$
- Inverse Elemente (I):
Für alle $a \in V$ existiert ein a' mit
 $a \bullet a' = n$ und $a + a' = e$

Beispiele: Mengenalgebra

- Beispiel 1: Mengenalgebra (T = Trägermenge)

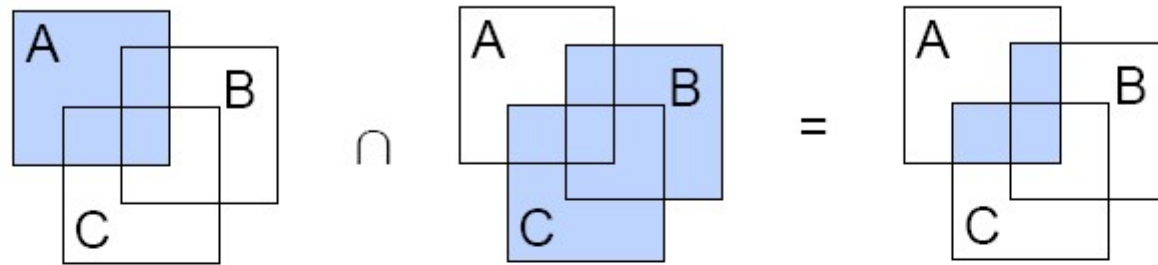
Boolesche Algebra	Mengenalgebra	
V	$\wp(T)$	Potenzmenge der Trägermenge T
\cdot	\cap	Durchschnitt
$+$	\cup	Vereinigung
n	\emptyset	Leere Menge
e	T	Trägermenge
a'	$T \setminus A$	Komplementärmenge

- Veranschaulichung durch Venn-Diagramme



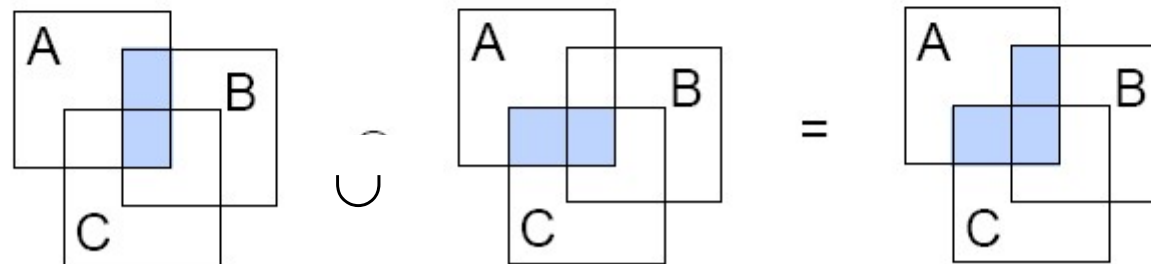
Mengenalgebra (2)

■ $A \cap (B \cup C)$



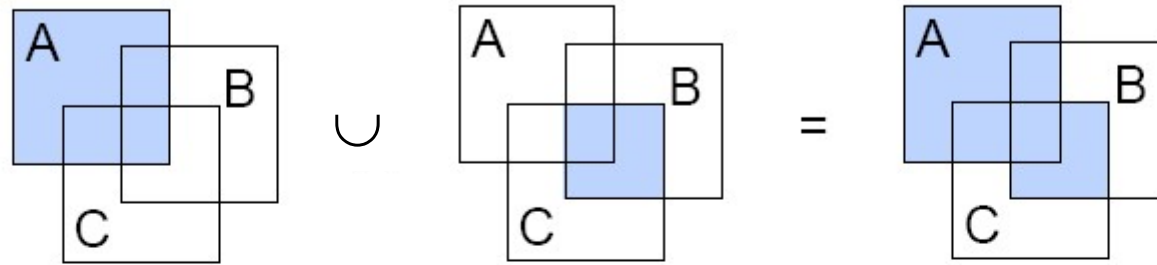
II

$(A \cap B) \cup (A \cap C)$



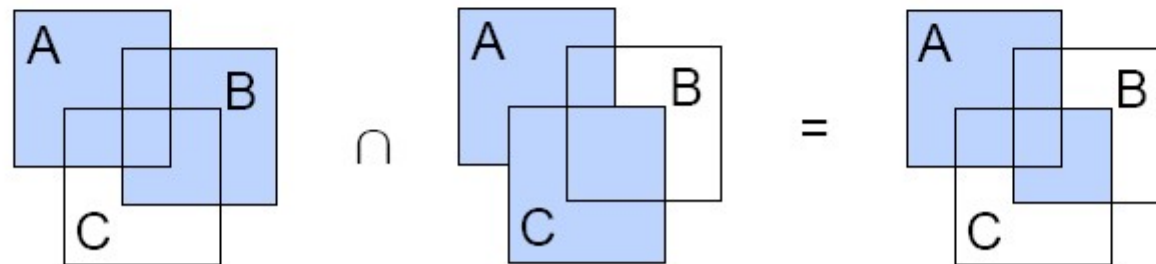
Mengenalgebra (3)

- $A \cup (B \cap C)$



II

- $(A \cup B) \cap (A \cup C)$



Beispiel: Schaltalgebra

■ Beispiel 2: Schaltalgebra

Boolesche Algebra	Schaltalgebra	
\vee	$\{1, 0\}$	Wahrheitswerte (TRUE, FALSE)
\bullet	\wedge	Konjunktion (UND-Operator)
$+$	\vee	Disjunktion (ODER-Operator)
n	0	„Falsch“ (FALSE)
e	1	„Wahr“ (TRUE)
a'	$\neg a$	Negation (Verneinung)

■ Konjunktion: $z = x \wedge y$

	x	y	z
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

■ Disjunktion: $z = x \vee y$

	x	y	z
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

■ Negation: $z = \neg x = \bar{x} = x'$

	x	z
0	0	1
1	1	0

Schaltalgebra und Axiome von Huntington

- Wahrheitstabellen für die Huntington'schen Axiome

				$x \wedge (y \vee z)$	$(x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z)$	$(x \vee y) \wedge (x \vee z)$	$x \wedge y$	$y \wedge x$	$x \vee y$	$y \vee x$	$x \wedge 1$	$x \vee 0$	$x \wedge \bar{x}$	$x \vee \bar{x}$
	z	y	x	D1	D2			K1		K2		N1	N2	I1	I2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1	0	0	1	1	1	1	0	1
2	0	1	0	0	0	0	0	0	0	1	1				
3	0	1	1	1	1	1	1	1	1	1	1				
4	1	0	0	0	0	0	0								
5	1	0	1	1	1	1	1								
6	1	1	0	0	0	1	1								
7	1	1	1	1	1	1	1								

- damit ist die Schaltalgebra eine Boolesche Algebra

Schaltalgebra

- **Schaltalgebra ist Spezialfall einer Booleschen Algebra**

- Schaltvariable: x

- kann die logischen Werte 0 (falsch) und 1 (wahr) annehmen

- Verknüpfungen

- UND, ODER, NICHT

- logische Schreibweise: $x \wedge y$, $x \vee y$, $\neg x$

- algebraische Schreibweise $x \cdot y$, $x + y$, \bar{x} oder x'

- algebraische Schreibweise ist kompakter

- » Punkt kann weggelassen werden

- » Punktrechnung geht vor Strichrechnung, dadurch entfallen viele Klammern

- Rechenregeln für Schaltvariablen

- Huntington'sche Axiome sind erfüllt (s.u.)

- daher gelten auch alle weiteren Rechenregeln der Booleschen Algebra

übersichtlicher

einfacher zu setzen

Boolesche Algebra, Rechenregeln

	Disjunktion	Konjunktion
⇒ Neutrales Element	$a+0 = a$	$a \cdot 1 = a$
⇒ Inverses Element	$a+a' = 1$	$a \cdot a' = 0$
⇒ Kommutativgesetz	$a+b = b+a$	$a \cdot b = b \cdot a$
Assoziativgesetz	$(a+b)+c = a+(b+c)$	$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
⇒ Distributivgesetz	$a \cdot (b+c) = a \cdot b + a \cdot c$	$a+b \cdot c = (a+b) \cdot (a+c)$
Eliminationsgesetz	$a+1 = 1$	$a \cdot 0 = 0$
Idempotenz	$a+a = a$	$a \cdot a = a$
Doppelte Negation	$a'' = a$	
Absorption	$a \cdot (a+b) = a$	$a+a \cdot b = a$
De-Morgan-Regel	$(a+b)' = a' \cdot b'$	$(a \cdot b)' = a' + b'$

die Zeilen 1-3 und 5 (⇒) definieren die Boolesche Algebra nach Huntington, der Rest folgt daraus

Beispiel: Idempotenzgesetze

- **Idempotenzgesetze**
 - sind wichtig für die Vereinfachung von Schaltfunktionen

Idempotenzgesetze

Herleitung von (ID1):

$$(ID1) \quad x \vee x = x$$

$$(ID2) \quad x \wedge x = x$$

$$x \vee x$$

$$= (x \vee x) \wedge 1$$

$$= (x \vee x) \wedge (x \vee \bar{x})$$

$$= x \vee (x \wedge \bar{x})$$

$$= x \vee 0$$

$$= x$$

Neutrales Element

Inverses Element

Distributivgesetz

Inverses Element

Neutrales Element

Achtung: Fehler im Buch!
Wo steckt der Fehler?

Beispiel: Absorptionsgesetze

- **Absorptionsgesetze**
 - sind wichtig für die Vereinfachung von Schaltfunktionen

Absorptionsgesetze

$$(AB1) \quad x \vee (x \wedge y) = x$$

$$(AB2) \quad x \wedge (x \vee y) = x$$

Herleitung von (AB1):

$$\begin{aligned} & x \vee (x \wedge y) \\ &= (x \wedge 1) \vee (x \wedge y) \\ &= x \wedge (1 \vee y) \\ &= x \wedge (y \vee 1) \\ &= x \wedge 1 \\ &= x \end{aligned}$$

Beispiel: Absorptionsgesetze (2)

- **Tipp (funktioniert aber nur bei Schaltalgebra)**
 - wenn man die Absorptionsregel (oder irgendeine andere Regel) vergessen hat, kommt man oft mit einer Fallunterscheidung schneller ans Ziel:

- 1. Fall $y = 0$, dann gilt:

$$\begin{aligned}x \vee (x \wedge y) &= x \vee (x \wedge 0) \\&= x \vee 0 \\&= x\end{aligned}$$

- 2. Fall $y = 1$, dann gilt:

$$\begin{aligned}x \vee (x \wedge y) &= x \vee (x \wedge 1) \\&= x \vee x \\&= x\end{aligned}\quad \text{(wenn man das schon weiß)}$$

Beispiel: Assoziativgesetz

Assoziativgesetze

$$(A1) \quad x \vee (y \vee z) = (x \vee y) \vee z$$

$$(A2) \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

Herleitung von (A1):

$$\begin{aligned} & x \vee (y \vee z) \\ &= (x \vee (y \vee z)) \wedge 1 \\ &= (x \vee (y \vee z)) \wedge (x \vee \bar{x}) \\ &= [(x \vee (y \vee z)) \wedge x] \vee [(x \vee (y \vee z)) \wedge \bar{x}] \\ &= [x] \vee [(x \vee (y \vee z)) \wedge \bar{x}] \\ &= [x \vee (x \wedge z)] \vee [(x \vee (y \vee z)) \wedge \bar{x}] \\ &= [(x \wedge (x \vee y)) \vee (x \wedge z)] \vee [(x \vee (y \vee z)) \wedge \bar{x}] \\ &= [x \wedge ((x \vee y) \vee z)] \vee [(x \vee (y \vee z)) \wedge \bar{x}] \\ &= [((x \vee y) \vee z) \wedge x] \vee [(x \vee (y \vee z)) \wedge \bar{x}] \\ &= [((x \vee y) \vee z) \wedge x] \vee [\bar{x} \wedge (x \vee (y \vee z))] \\ &= [((x \vee y) \vee z) \wedge x] \vee [(\bar{x} \wedge x) \vee (\bar{x} \wedge (y \vee z))] \\ &= [((x \vee y) \vee z) \wedge x] \vee [0 \vee (\bar{x} \wedge (y \vee z))] \\ &= [((x \vee y) \vee z) \wedge x] \vee [\bar{x} \wedge (y \vee z)] \\ &= [((x \vee y) \vee z) \wedge x] \vee [(\bar{x} \wedge y) \vee (\bar{x} \wedge z)] \\ &= [((x \vee y) \vee z) \wedge x] \vee [(0 \vee (\bar{x} \wedge y)) \vee (\bar{x} \wedge z)] \\ &= [((x \vee y) \vee z) \wedge x] \vee [((\bar{x} \wedge x) \vee (\bar{x} \wedge y)) \vee (\bar{x} \wedge z)] \\ &= [((x \vee y) \vee z) \wedge x] \vee [(\bar{x} \wedge (x \vee y)) \vee (\bar{x} \wedge z)] \\ &= [((x \vee y) \vee z) \wedge x] \vee [\bar{x} \wedge ((x \vee y) \vee z)] \\ &= [((x \vee y) \vee z) \wedge x] \vee [((x \vee y) \vee z) \wedge \bar{x}] \\ &= ((x \vee y) \vee z) \wedge [x \vee \bar{x}] \\ &= ((x \vee y) \vee z) \wedge 1 \\ &= (x \vee y) \vee z \end{aligned}$$

Schaltfunktionen

- **Definition Schaltfunktion**

- seien $B = \{0, 1\}$; $n, m \in \mathbb{N}$; $n, m \geq 1$
- eine Funktion $f: B^n \rightarrow B^m$ heißt Schaltfunktion



Boolesche Funktionen

- **Definition Boolesche Funktionen**

- eine Schaltfunktion mit $m=1$ nennt man auch $(n\text{-stellige})$ Boolesche Funktion

- **Offenbar gilt:**

- Eine Schaltfunktion mit m Ausgängen ist darstellbar als m Schaltfunktionen mit je einem Ausgang, also m Booleschen Funktionen:

$$f: B^n \rightarrow B^m \text{ entspricht } \left\{ \begin{array}{l} f_1: B^n \rightarrow B \\ \dots \\ f_m: B^n \rightarrow B \end{array} \right.$$

Wahrheitstabellen

- **Wahrheitstabelle**

- auch Wahrheitstafel, Wertetabelle, Wertetafel genannt
- jede n -stellige Boolesche Funktion lässt sich eindeutig als Wahrheitstabelle mit 2^n Zeilen darstellen

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

- jede der Zeilen kann entweder 0 oder 1 enthalten, also gibt es $2^{(2^n)}$ verschiedene n -stellige Boolesche Funktionen

2-stellige Boolesche Funktionen

Benennung der Verknüpfung	Funktionswert		Schreibweise mit den Zeichen $\wedge \vee -$	Bemerkung
	y	$= f(x_1, x_2)$		
	x_1	$=$		
	x_2	$=$		
Null	y_0	$=$	0	Null
UND-Verknüpfung	y_1	$=$	$x_1 \wedge x_2$	x_1 UND x_2
Inhibition	y_2	$=$	$\bar{x}_1 \wedge x_2$	
Transfer	y_3	$=$	x_2	
Inhibition	y_4	$=$	$x_1 \wedge \bar{x}_2$	
Transfer	y_5	$=$	x_1	
Antivalenz	y_6	$=$	$(x_1 \wedge \bar{x}_2) \vee (\bar{x}_1 \wedge x_2)$	Exklusiv-ODER
ODER-Verknüpfung	y_7	$=$	$x_1 \vee x_2$	x_1 ODER x_2
NOR-Verknüpfung	y_8	$=$	$\overline{x_1 \vee x_2}$	NICHT-ODER
Äquivalenz	y_9	$=$	$(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge \bar{x}_2)$	
Komplement	y_{10}	$=$	\bar{x}_1	
Implikation	y_{11}	$=$	$\bar{x}_1 \vee x_2$	
Komplement	y_{12}	$=$	\bar{x}_2	
Implikation	y_{13}	$=$	$x_1 \vee \bar{x}_2$	
NAND-Verknüpfung	y_{14}	$=$	$\overline{x_1 \wedge x_2}$	NICHT-UND
Eins	y_{15}	$=$	1	Eins

Boolesche Funktionen

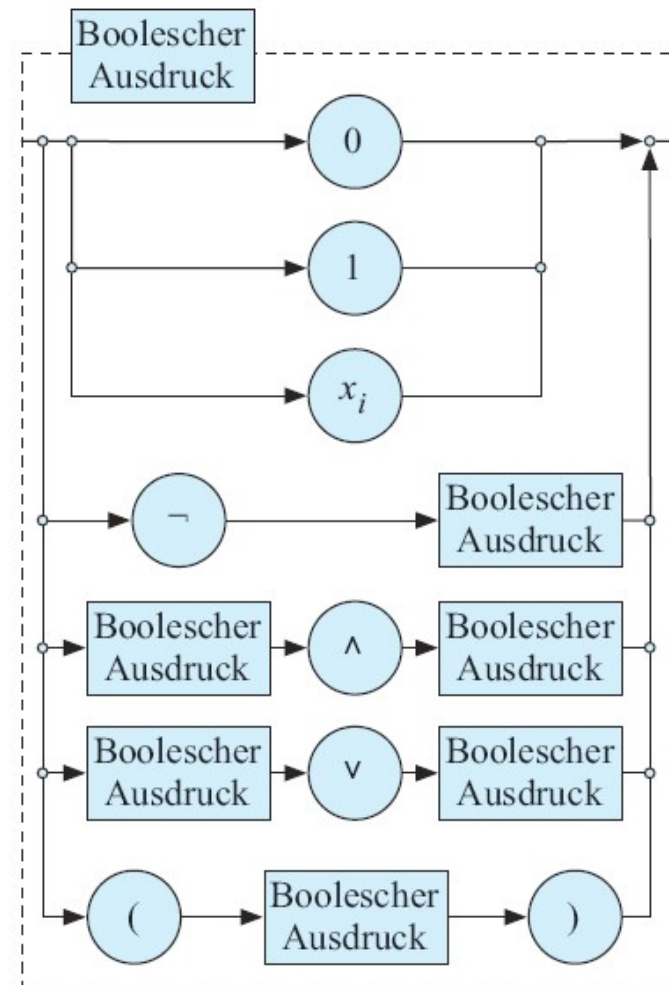
- **Darstellungsmöglichkeiten**
 - Boolesche Ausdrücke
 - "Formeln": Zeichenfolge mit
 - Konstanten 0, 1
 - Schaltvariable x_i
 - Boolesche Operatoren \neg , \wedge , \vee
 - Klammern (,)
 - Wahrheitstafel
 - alle Belegungen der freien Variablen werden explizit aufgezählt
 - Anzahl der Zeilen in der Wahrheitstabelle steigt exponentiell mit der Anzahl der freien Variablen
 - Graphen
 - dienen häufig der Computer-internen Repräsentation
 - z.B. Binäre Entscheidungsdiagramme
 - detaillierte Beschreibung s.u.

Boolesche Ausdrücke

- **Formale (rekursive) Definition**

- Sei $V = \{x_1, \dots, x_n\}$ eine Menge Boolescher Variablen.
- Die Menge aller Booleschen Ausdrücke definieren wir folgendermaßen:
 - $0, 1, x_i$ sind Boolesche Ausdrücke
 - Mit Φ ist auch $\neg\Phi$ ein Boolescher Ausdruck.
 - Mit Φ und Ψ sind auch $\Phi \wedge \Psi$ und $\Phi \vee \Psi$ Boolesche Ausdrücke.
 - Mit Φ ist auch (Φ) ein Boolescher Ausdruck.
- Damit ist z.B. $(x_1 \vee 0) \wedge (1 \wedge ((\neg x_2 \wedge x_3)))$ ein Boolescher Ausdruck.

Syntaxdiagramm:



Boolesche Ausdrücke (2)

- **Operatoren-Bindung**

- aus obigem Syntax-Diagramm ist die Bindung der Operatoren nicht herauszulesen

$$\neg x_2 \wedge x_3 = \neg(x_2 \wedge x_3) \quad ?$$
$$= (\neg x_2) \wedge x_3 \quad ?$$

- erst zusätzliche Regeln machen die Syntax eindeutig

- NICHT-Operator bindet stärker als UND- oder ODER-Operator.

- also

$$\neg x_2 \wedge x_3 = (\neg x_2) \wedge x_3$$

- Punktrechnung geht vor Strichrechnung.

- also

$$x_2 \cdot x_3 + x_2 \cdot x_4 = (x_2 \cdot x_3) + (x_2 \cdot x_4) = x_2 \cdot x_3 + x_2 x_4$$

- Aber stimmt auch "UND-Operator bindet stärker als ODER-Operator"?

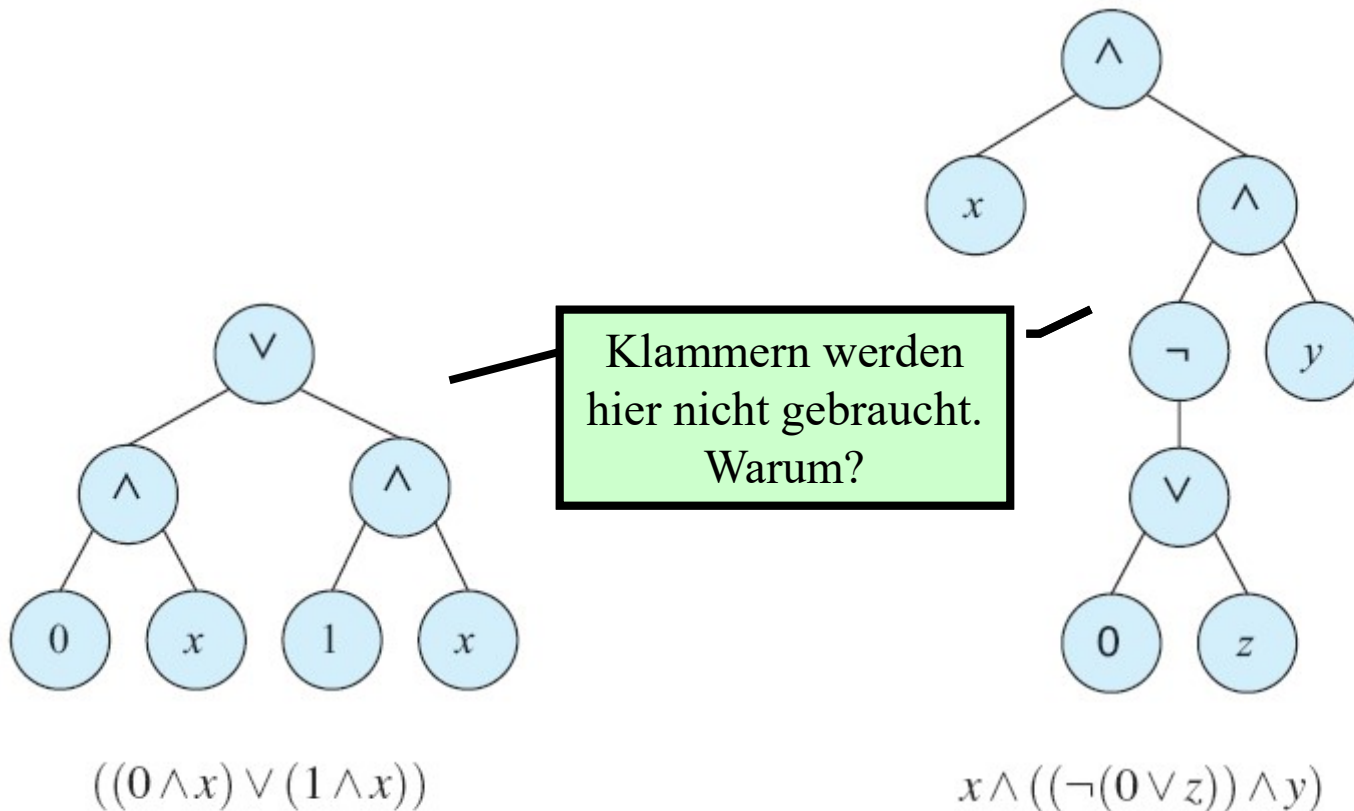
$$\neg x_2 \wedge x_3 \vee x_2 \wedge x_4 = (\neg x_2 \wedge x_3) \vee (x_2 \wedge x_4) \quad ?$$
$$= \neg x_2 \wedge (x_3 \vee x_2) \wedge x_4 \quad ?$$

Da würde ich mich nicht drauf verlassen. Besser die Klammern setzen!

den Punkt darf man
sogar weglassen

Boolesche Ausdrücke (3)

- Beispiele Boolescher Ausdrücke in Baumdarstellung



Boolesche Ausdrücke (3)

- Sei Φ ein beliebiger Boolescher Ausdruck.
- Φ heißt
 - *erfüllbar*, wenn es Werte x_1, \dots, x_n gibt, mit $\Phi(x_1, \dots, x_n) = 1$.
 - *unerfüllbar*, wenn Φ stets zu 0 evaluiert.
 - *allgemeingültig*, wenn Φ stets zu 1 evaluiert.
 - Einen allgemeingültigen Ausdruck bezeichnet man auch als *Tautologie*.

Erfüllbare Funktionen

$$\begin{aligned}\phi_1 &= \neg x \\ \phi_2 &= x \wedge y \\ \phi_3 &= x \vee y\end{aligned}$$

Unerfüllbare Funktionen

$$\begin{aligned}\phi_1 &= 0 \\ \phi_2 &= x \wedge \neg x \\ \phi_3 &= \neg(x \vee \neg x)\end{aligned}$$

Allgemeingültige Funktionen

$$\begin{aligned}\phi_1 &= 1 \\ \phi_2 &= x \vee \neg x \\ \phi_3 &= \neg(x \wedge \neg x)\end{aligned}$$

Äquivalenz Boolescher Funktionen

- **Definition**

- Zwei Boolesche Ausdrücke Φ und Ψ sind genau dann äquivalent, wenn sie dieselbe Funktion repräsentieren, d.h. wenn für alle Variablenbelegungen x_1, \dots, x_n gilt:

$$\Phi(x_1, \dots, x_n) = \Psi(x_1, \dots, x_n)$$

- **Anders gesagt**

- Zwei Boolesche Ausdrücke Φ und Ψ sind genau dann äquivalent, wenn der Ausdruck $\Phi \leftrightarrow \Psi$ eine Tautologie ist.
- Hierbei ist der Äquivalenz-Operator „ \leftrightarrow “ definiert als:

$$z = x \leftrightarrow y$$

x	y	z
0	0	1
0	1	0
1	0	0
1	1	1

oder

Äquivalenz		
$x \leftrightarrow y$	=	$(\bar{x} \wedge \bar{y}) \vee (x \wedge y)$
	=	$(\bar{x} \vee y) \wedge (x \vee \bar{y})$

Feststellen der Äquivalenz

- **Wie stellt man fest, ob Φ und Ψ äquivalent sind?**
 - Vergleich der Wahrheitstabellen
 - alle Belegungen aufzählen (für beide Funktionen in derselben Reihenfolge)
 - Vergleich der Funktionswertspalte
 - geht in der Praxis nur bei einfachen Funktionen mit wenigen Variablen
 - Algebraische Umformung
 - sukzessive Anwendung von Rechenregeln, um Φ in Ψ zu überführen oder um $\Phi \leftrightarrow \Psi$ zu 1 zu vereinfachen
 - Erzeugen einer Normalform
 - Transformation der Ausdrücke Φ und Ψ in eine Normalform
 - Normalform bedeutet, dass die Darstellung eindeutig ist
 - die Ausdrücke Φ und Ψ müssen dann dieselbe Darstellung haben
 - z.B. ist die Wahrheitstabelle eine Normalform
 - es gibt weitere Normalformen (z.B. disjunktive und konjunktive Normalform, diverse Entscheidungsdiagramme, etc.) (s.u.)

Strukturelle Induktion

- **Spezielle Variante der vollständigen Induktion**

- Induktionsanfang

- Gültigkeit der Behauptung wird zunächst bewiesen für einen oder mehrere einfache Basisfälle

- Elementare Wahrheitswerte 0 und 1 und alle booleschen Ausdrücke, die nur aus einer Variablen x_i bestehen

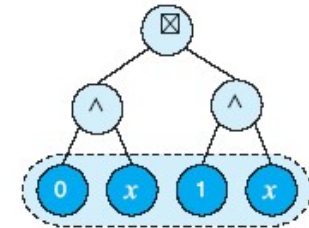
- (wegen $0 = x \wedge \neg x$ und $1 = x \vee \neg x$ reicht es streng genommen sogar aus, nur die Variablen x_i zu betrachten)

- Induktionsschritt

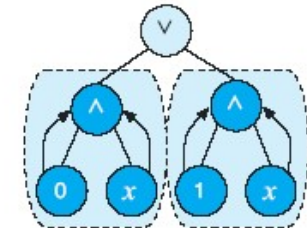
- Übertragung der Behauptung auf das nächst komplexere Objekt

- zusammengesetzte Ausdrücke
- Gültigkeit kann angenommen werden für alle Teilausdrücke, die ja einfacher sind

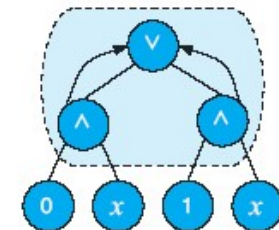
- Induktionsanfang



- Induktionsschluss



- Erneuter Induktionsschluss



Strukturelle Induktion (2)

- **Strukturelle Induktion**
 - entspricht klassischer Induktion über die Formellänge n
- **Formellänge n**
 - Anzahl der Symbole in der Formel (Knoten im Baum, d.h. 0, 1, Variablen, Operatoren)
 - 0, 1, x_i sind die einzigen Ausdrücke der Länge 1
 - Induktionsanfang beweist die Behauptung für $n=1$
 - jedes weitere Symbol vergrößert die Formellänge
 - um die Behauptung für $n>1$ zu zeigen, darf man annehmen, dass die Behauptung bereits für alle Formellängen bis maximal $n-1$ gezeigt wurde

Negationstheorem

Sei $f(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg)$ ein boolescher Ausdruck, in dem neben den Konstanten 1 und 0 und den Variablen x_1, \dots, x_n die booleschen Operatoren \wedge, \vee und \neg vorkommen. Dann gilt:

$$\overline{f(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg)} = f(1, 0, \overline{x_1}, \dots, \overline{x_n}, \vee, \wedge, \neg)$$

- d.h. einen beliebig komplexen Booleschen Ausdruck mit den Grundoperatoren kann man dadurch negieren, dass man folgende Ersetzungen vornimmt:
 - 0 durch 1
 - 1 durch 0
 - \wedge durch \vee
 - \vee durch \wedge
 - alle Variablen durch die negierten Variablen
 - Klammern müssen evtl. gemäß den Bindungsregeln neu gesetzt werden

Negationstheorem (2)

- **Beweis durch strukturelle Induktion**

- Induktionsanfang: $n=1$

- 1. Fall: $f = 0$

$$\neg f(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) = \neg 0 = 1 = f(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg)$$

- 2. Fall: $f = 1$

$$\neg f(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) = \neg 1 = 0 = f(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg)$$

- 3. Fall: $f = x_i$

$$\neg f(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) = \neg(x_i) = (\neg x_i) = f(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg)$$

Negationstheorem (3)

- **Induktionsschritt:** Der boolesche Ausdruck f ist auf jeden Fall aus "kleineren" booleschen Ausdrücke f_1 und f_2 , für die Behauptung schon bewiesen wurde, unter Verwendung von Operatoren zusammengesetzt.

- 1. Fall: $f = \neg f_1$

$$\begin{aligned}\neg f(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) &= \neg \neg f_1(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) \\ &= \neg f_1(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg) \\ &= f(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg)\end{aligned}$$

- 2. Fall: $f = f_1 \wedge f_2$

$$\begin{aligned}\neg f(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) &= \neg(f_1(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) \wedge f_2(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg)) \\ &= \neg f_1(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) \vee \neg f_2(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) \\ &= f_1(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg) \vee f_2(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg) \\ &= f(1, 0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \neg)\end{aligned}$$

de Morgan

- 3. Fall: $f = f_1 \vee f_2$
– analog

Dualitätsprinzip der Booleschen Algebra

- **wichtige Schlussfolgerung**

Sei

$$\phi(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg) = \psi(0, 1, x_1, \dots, x_n, \wedge, \vee, \neg)$$

ein Gesetz der booleschen Algebra, in der neben Variablen und den Konstanten 0 und 1 ausschließlich die Elementarverknüpfungen \neg , \wedge und \vee vorkommen. Dann ist auch die *duale Gleichung*

$$\phi(1, 0, x_1, \dots, x_n, \vee, \wedge, \neg) = \psi(1, 0, x_1, \dots, x_n, \vee, \wedge, \neg)$$

ein Gesetz der booleschen Algebra.

- **Beweis**

- Negiere beide Seiten und wende Negationstheorem an.
- Ersetze alle \bar{x}_i durch x_i . (Wieso darf man das?)

Literale, Monome und Polynome

- **Definitionen**

- Literal

- Variable oder invertierte Variable
 - z.B. x_1 oder x_1'

- Monom

- Konjunktion (UND-Verknüpfung) von Literalen
 - z.B. $x_1x_2x_3'x_4$

- man muss nicht über Klammern die Reihenfolge der 2-stelligen UND-Verknüpfungen festlegen

- wegen des Assoziativgesetzes ist das Ergebnis eindeutig:

$$\begin{aligned}x_1x_2x_3'x_4 &= ((x_1x_2)x_3')x_4 \\ &= (x_1x_2)(x_3'x_4) = x_1(x_2(x_3'x_4)) = x_1((x_2x_3')x_4) = (x_1(x_2x_3'))x_4\end{aligned}$$

- Polynom

- Disjunktion (ODER-Verknüpfung) von Monomen
 - z.B. $x_1x_2x_3'x_4 + x_1'x_2x_3x_4'$

Minterme

- **vollständiges Monom**

- ein Monom heißt vollständig genau dann, wenn *alle* x_i mit $1 \leq i \leq n$ genau einmal in ihm vorkommen
- ein vollständiges Monom heißt auch **Minterm**
 - Wahrheitstabelle für Minterme beinhaltet nur eine einzige 1 als Funktionswert
 - "kleinste" Boolesche Funktion, die von 0 verschieden ist

- z.B. $x_1x_2'x_3$:

x_1	x_2	x_3	$x_1x_2'x_3$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Beachte: ein Minterm ist selbst eine Boolesche Funktion!

Disjunktive Normalform, DNF

- **vollständiges Polynom**

- ein Polynom heißt *vollständig* genau dann, wenn alle in ihm vorkommenden Monome vollständig sind (es also nur aus Mintermen besteht)

- **Satz**

- jede Boolesche Funktion lässt sich durch ein vollständiges Polynom darstellen
- diese Darstellung heißt "disjunktive Normalform" (DNF)
 - bis auf Permutationen innerhalb der Monome bzw. in der Reihenfolge der Disjunktionen ist die Darstellung eindeutig
 - Sortierung der Terme (z.B. wie in Wertetabelle): eindeutige Darstellung

- **Beweis**

- trivial, da jede 1 in der Wertetabelle durch einen entsprechenden Minterm erzeugt werden kann. Durch ODER-Verknüpfung dieser Minterme entsteht die gewünschte Wertetabelle.

Beispiel DNF

- Beispiel

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	Minterm	$x_1'x_2'x_3$	$x_1'x_2x_3'$	$x_1x_2'x_3$
0	0	0	0		0	0	0
0	0	1	1	$x_1'x_2'x_3$	1	0	0
0	1	0	1	$x_1'x_2x_3'$	0	1	0
0	1	1	0		0	0	0
1	0	0	0		0	0	0
1	0	1	1	$x_1x_2'x_3$	0	0	1
1	1	0	0		0	0	0
1	1	1	0		0	0	0

$$f(x_1, x_2, x_3) = x_1'x_2'x_3 + x_1'x_2x_3' + x_1x_2'x_3$$

Konjunktive Normalform, KNF

- es geht auch umgekehrt
 - jede Boolesche Funktion lässt sich als konjunktive Normalform schreiben, also als Konjunktion von Maxtermen (vollständige Disjunktionen, d.h. genau eine 0 in der Wahrheitstabelle)
 - z.B.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	Maxterm	$x_1+x_2'+x_3$	$x_1+x_2'+x_3'$	$x_1'+x_2+x_3'$
0	0	0	1		1	1	1
0	0	1	1		1	1	1
0	1	0	0	$x_1+x_2'+x_3$	0	1	1
0	1	1	0	$x_1+x_2'+x_3'$	1	0	1
1	0	0	1		1	1	1
1	0	1	0	$x_1'+x_2+x_3'$	1	1	0
1	1	0	1		1	1	1
1	1	1	1		1	1	1

$$f(x_1, x_2, x_3) = (x_1+x_2'+x_3) (x_1+x_2'+x_3') (x_1'+x_2+x_3')$$

Normalformen

- jede Schaltfunktion kann als DNF oder KNF dargestellt werden
 - DNF ist einfacher, wenn die Wertetabelle wenige 1'en enthält
 - KNF ist einfacher, wenn die Wertetabelle wenige 0'en enthält
- häufig enthalten Wertetabellen aber sehr viele 1'en und 0'en (je etwa zur Hälfte), so dass diese Normalform-Darstellungen sehr umfangreich sind

Vollständige Operatorensysteme

- **Definition**

- Sei M eine beliebige Menge von Operatoren.
- M ist ein **vollständiges Operatorensystem**, wenn sich jede Boolesche Funktion durch einen Ausdruck beschreiben lässt, in dem neben den Variablen x_1, \dots, x_n ausschließlich Operatoren aus M vorkommen.
 - Braucht man neben den eigentlichen Operatoren noch die Werte 0 oder 1, gehören diese ebenfalls in die Menge M .

- **Beispiele für vollständige Operatorensysteme**

- jede Boolesche Funktion lässt sich mithilfe von \neg, \wedge, \vee beschreiben (siehe z.B. Darstellung als DNF), daher ist $\{\neg, \wedge, \vee\}$ ein vollständiges Operatorensystem.
- Aber auch NAND oder NOR alleine bilden schon ein vollständiges Operatorensystem:

Vollständige Operatorensysteme (2)

- Definition von NAND und NOR

$$z = x \overline{\wedge} y$$

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

Sheffer-Funktion
(NAND)

$$z = x \overline{\vee} y$$

x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

Peirce-Funktion
(NOR)

- Reduktion der Grundoperationen auf NAND und NOR

Reduktion von \wedge , \vee , \neg auf NAND

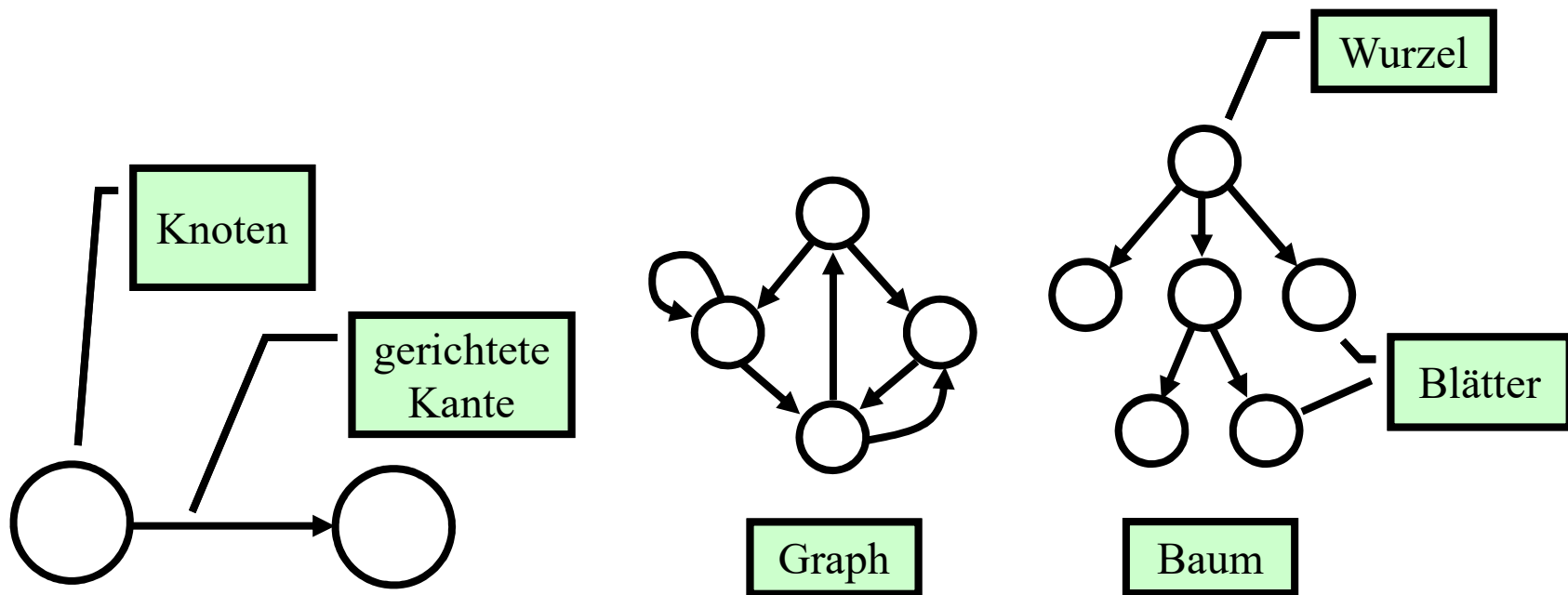
$$\begin{aligned}\bar{x} &= (\overline{x \wedge x}) \\ x \wedge y &= \overline{\overline{x \wedge y}} \\ &= \overline{\overline{x \wedge y} \wedge \overline{x \wedge y}} \\ x \vee y &= \overline{\overline{x \vee y}} \\ &= \overline{\overline{x} \wedge \overline{y}} \\ &= \overline{\overline{x \wedge x} \wedge \overline{y \wedge y}}\end{aligned}$$

Reduktion von \wedge , \vee , \neg auf NOR

$$\begin{aligned}\bar{x} &= (\overline{x \vee x}) \\ x \wedge y &= \overline{\overline{x \wedge y}} \\ &= \overline{\overline{x} \vee \overline{y}} \\ &= \overline{\overline{x \vee x} \vee \overline{y \vee y}} \\ x \vee y &= \overline{\overline{x \vee y}} \\ &= \overline{\overline{x \vee y} \vee \overline{x \vee y}}\end{aligned}$$

Binäre Entscheidungsdiagramme

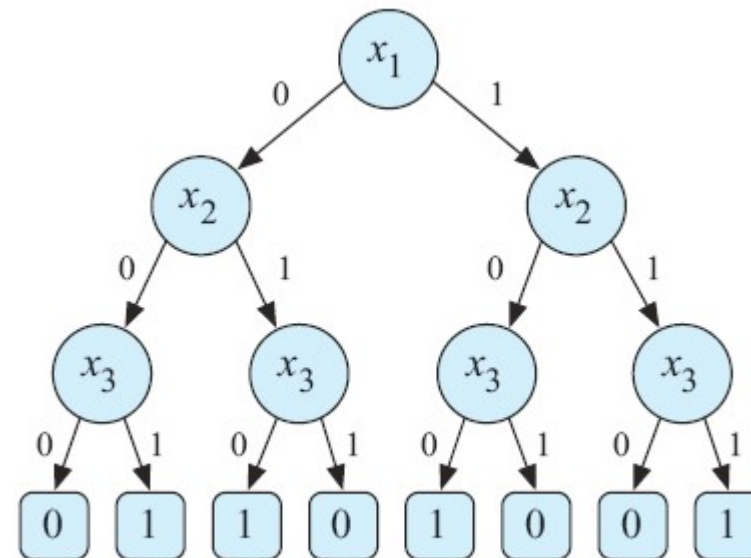
- **Binary Decision Diagrams, BDD**
 - relativ junge Entwicklung (erste Veröffentlichungen 1958)
 - auf Graphen basierende Repräsentation Boolescher Funktionen
- **Begriffe zu gerichteten Graphen**



Binäre Entscheidungsdiagramme (2)

- **Binary Decision Diagrams, BDD**
 - jeder Weg von der Wurzel zu einem Blatt stellt eine Variablenbelegung dar
 - der Funktionswert steht als Markierung in dem Blatt
- **Beispiel 3-stellige Paritätsfunktion**

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



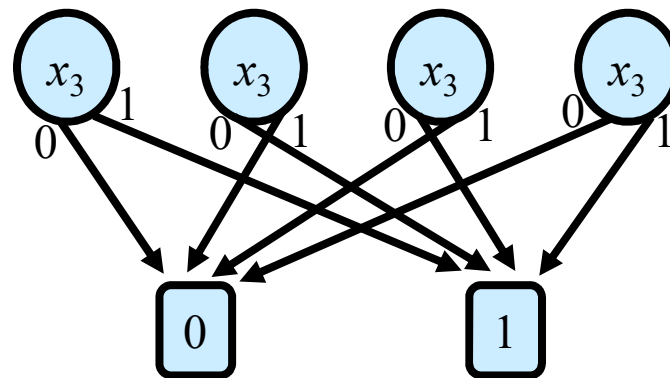
Geordnete binäre Entscheidungsdiagramme

- **Ordered Binary Decision Diagram, OBDD**

- Eine binärer Entscheidungsbaum heißt *geordnet*, wenn die Reihenfolge der Variablen auf allen Pfaden von der Wurzel zu den Blättern gleich ist.
- obiges Beispiel ist schon ein OBDD

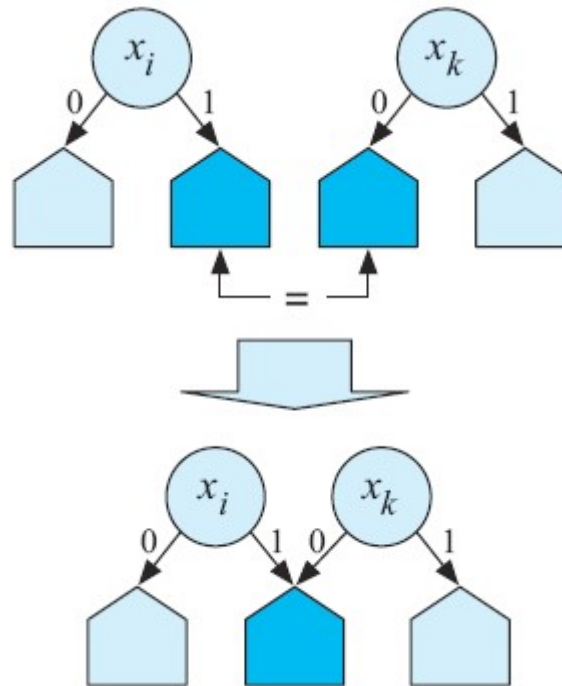
- **Redundanzen**

- Baum kann erheblich vereinfacht werden
 - Überführung des Baumes in einen Graphen durch Verwendung von nur 2 Blättern zur Darstellung der 0 und der 1



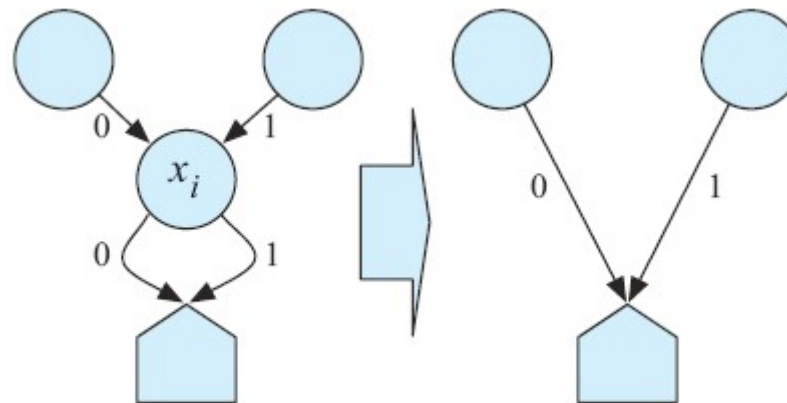
Vereinfachungsregeln

- **Verschmelzen gleicher Teilgraphen**
 - angefangen bei den Blättern werden in jeden Schritt identische Teilgraphen verschmolzen
 - zwei Teilgraphen sind genau dann identisch, wenn deren linke und rechte Kanten alle jeweils zu den gleichen Nachfolgeknoten zeigen



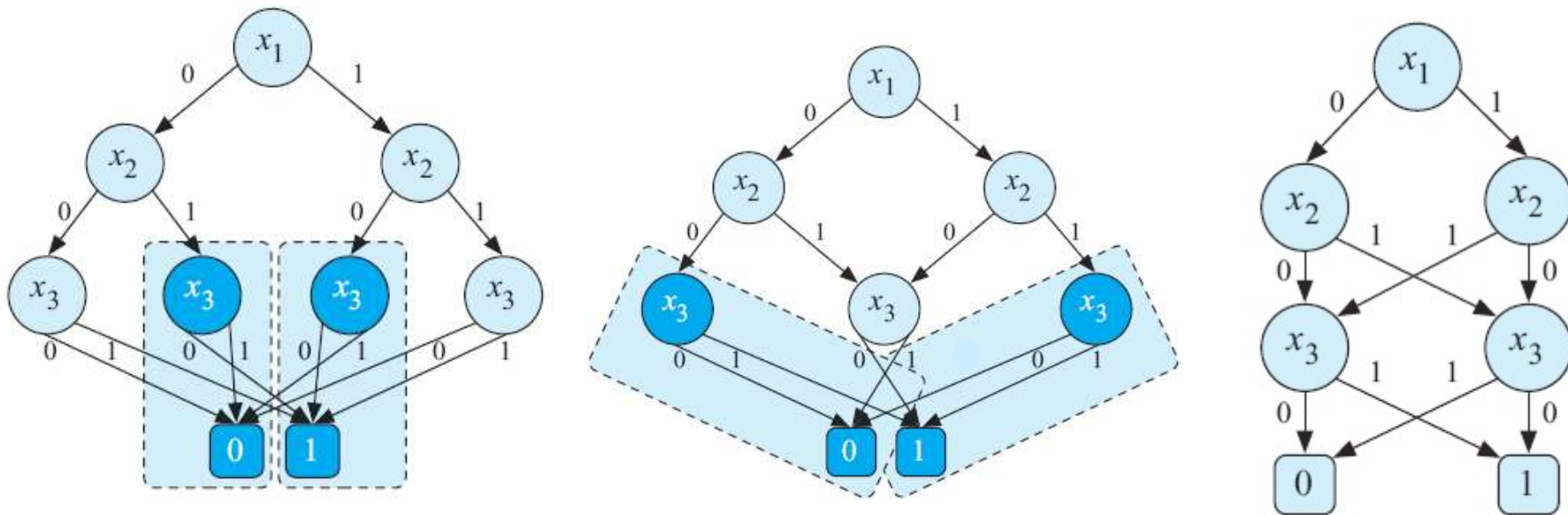
Vereinfachungsregeln (2)

- **Löschen von Knoten mit zwei gleichen Nachfolgern**
 - beim Verschmelzen entstehen gelegentlich Knoten, deren beiden ausgehenden Kanten auf denselben Nachfolgeknoten verweisen
 - der Wert der entsprechenden Variablen ist damit unerheblich
 - der Knoten wird gelöscht (Resolutionsregel, s.u.)
 - die Kanten, die auf den gelöschten Knoten zeigten, werden auf den Nachfolgeknoten umgesetzt



Reduziertes geordnetes Entscheidungsdiagramm

- **Reduced Ordered Binary Decision Diagram, ROBDD**
 - Ein geordnetes Entscheidungsdiagramm heißt *reduziert*, wenn keine Vereinfachungsregel mehr anwendbar ist.
- **Beispiel 3-stellige Paritätsfunktion**
 - nach zwei Schritten entsteht schon ein ROBDD



Reduziertes geordnetes Entscheidungsdiagramm (2)

- **Wichtiger Satz von Randal Bryant (1986)**
 - Binäre Entscheidungsäume, die sowohl geordnet als auch reduziert sind, stellen jede Boolesche Funktion eindeutig dar.
 - Mit anderen Worten:

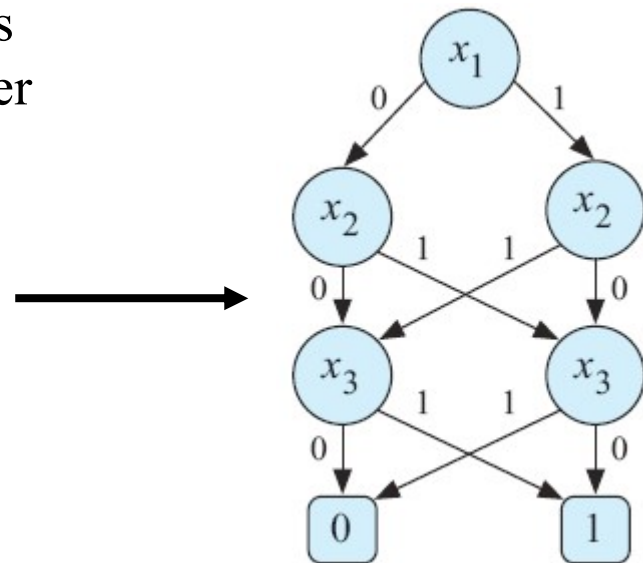
ROBDDs sind eine weitere Normalform Boolescher Funktionen.

- ROBDDs werden häufig zur Äquivalenzfeststellung von Booleschen Funktionen verwendet

Reduziertes geordnetes Entscheidungsdiagramm (3)

- **Größe von ROBDDs**

- die Wahrheitstabelle wächst immer exponentiell mit der Anzahl der Variablen
- viele Formelklassen können als ROBDDs kompakter dargestellt werden als mit einer Wertetabelle
- z.B. steigt die Anzahl der Knoten für die n -stellige Paritätsfunktion nur linear mit der Anzahl n der Variablen
- im Allgemeinen wächst aber auch die Knotenzahl von ROBDDs exponentiell mit der Anzahl der Variablen



Entwicklungssatz von Shannon

Sei f eine beliebige n -stellige boolesche Funktion. Dann gilt:

$$f(x_1, \dots, x_n) = (x_i \wedge f_{x_i=1}) \vee (\bar{x}_i \wedge f_{x_i=0})$$

$f_{x_i=1}$ und $f_{x_i=0}$ bezeichnen den *positiven* und den *negativen Kofaktor* von f und sind wie folgt definiert:

$$f_{x_i=1} := f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

$$f_{x_i=0} := f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

- **Beweis durch Fallunterscheidung**

- Für $x_i = 1$ ist die Behauptung äquivalent zu

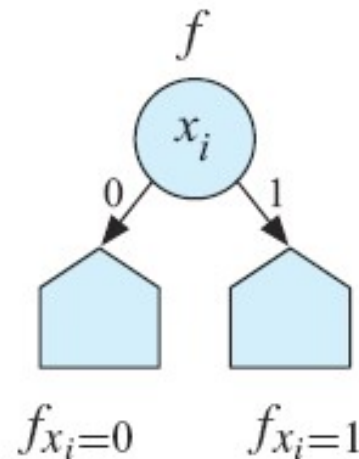
$$f(x_1, \dots, x_{i-1}, x_i = 1, x_{i+1}, \dots, x_n) = (1 \wedge f_{x_i=1}) \vee (0 \wedge f_{x_i=0}) = f_{x_i=1}$$

- was wegen der Definition des Kofaktors korrekt ist.
- Für $x_i = 0$ gilt ebenso:

$$f(x_1, \dots, x_{i-1}, x_i = 0, x_{i+1}, \dots, x_n) = (0 \wedge f_{x_i=1}) \vee (1 \wedge f_{x_i=0}) = f_{x_i=0}$$

Zusammenhang BDD und Boolesche Funktion

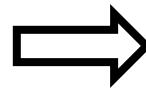
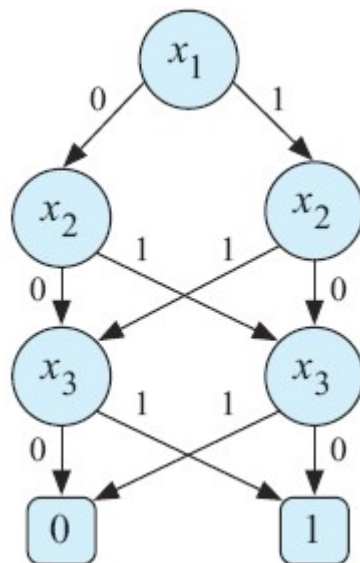
- Die Kofaktoren sind nichts weiter, als die Ergebnisse der Fallunterscheidung bzgl. einer Variablen.
- Der Entwicklungssatz von Shannon zeigt, wie man die Originalfunktion f aus ihren Kofaktoren rekonstruieren kann.
- In einem BDD entspricht der linke Nachfolger eines Knotens x_i dem negativen Kofaktor bzgl. x_i und der rechte Nachfolger dem positiven Kofaktor:



$$f(x_1, \dots, x_n) = (x_i \wedge f_{x_i=1}) \vee (\bar{x}_i \wedge f_{x_i=0})$$

Beispiel 3-stellige Paritätsfunktion

- Entwicklungssatz von Shannon rekursiv auf alle Knoten anwenden
- damit kann man ein BDD in eine Boolesche Funktion umformen



$$\begin{aligned}
 y &= (x_1 \wedge f_{x_1=1}) \vee (\bar{x}_1 \wedge f_{x_1=0}) \\
 &= \dots \\
 &= (x_1 \wedge ((x_2 \wedge ((x_3 \wedge 1) \vee (\bar{x}_3 \wedge 0))) \vee (\bar{x}_2 \wedge ((x_3 \wedge 0) \vee (\bar{x}_3 \wedge 1))))) \vee \\
 &\quad (\bar{x}_1 \wedge ((x_2 \wedge ((x_3 \wedge 0) \vee (\bar{x}_3 \wedge 1))) \vee (\bar{x}_2 \wedge ((x_3 \wedge 1) \vee (\bar{x}_3 \wedge 0))))) \\
 &= (x_1 \wedge ((x_2 \wedge x_3) \vee (\bar{x}_2 \wedge \bar{x}_3))) \vee \\
 &\quad (\bar{x}_1 \wedge ((x_2 \wedge \bar{x}_3) \vee (\bar{x}_2 \wedge x_3)))
 \end{aligned}$$

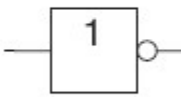
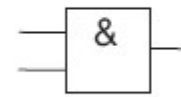
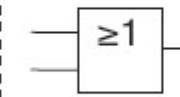
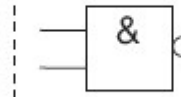
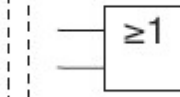
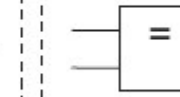
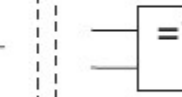






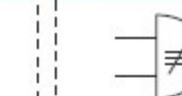

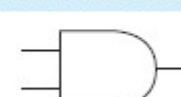



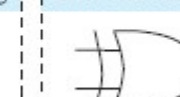
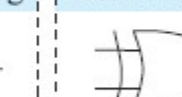
Logikgatter

- Schaltzeichen verschiedener Logikgatter

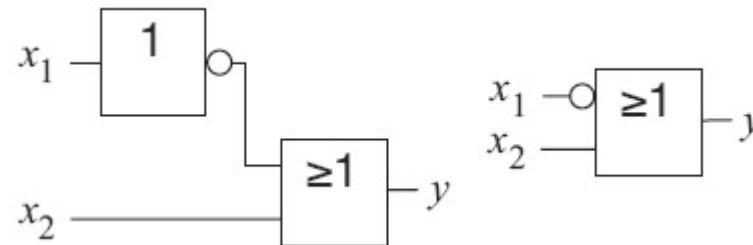
NICHT-Gatter
(Inverter)

UND-
Gatter

ODER-
Gatter

Negation	Konjunktion	Disjunktion	NAND	NOR	Äquivalenz	Antivalenz
						
DIN 40900	DIN 40900	DIN 40900	DIN 40900	DIN 40900	DIN 40900	DIN 40900
						
alte Darstellung	alte Darstellung	alte Darstellung	alte Darstellung	alte Darstellung	alte Darstellung	alte Darstellung
						
US-Norm	US-Norm	US-Norm	US-Norm	US-Norm	US-Norm	US-Norm

verkürzte Notation der Negation:



Schaltnetze

- **Schaltnetz**
 - technische Realisierung einer Schaltfunktion mithilfe von Gattern
 - jede Schaltfunktion kann als DNF dargestellt werden
 - damit kann auch jede Schaltfunktion als Schaltnetz, z.B. mit UND-, ODER- und NICHT-Gattern realisiert werden
- **Vorgehensweise zur Aufstellung der DNF**
 - Wertetabelle aufstellen
 - alle Zeilen suchen, die eine 1 ergeben
 - zugehörige Minterme aufstellen
 - Minterme durch Disjunktion zu Polynomen zusammenfassen
- **Beispiel: Majorität dreier Schaltvariablen**
 - Ausgabe ist 1, wenn mindestens 2 Eingänge 1 sind

Beispiel Majorität

Wertetabelle:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	Minterm	$x_1'x_2x_3$	$x_1x_2'x_3$	$x_1x_2x_3'$	$x_1x_2x_3$
0	0	0	0		0	0	0	0
0	0	1	0		0	0	0	0
0	1	0	0		0	0	0	0
0	1	1	1	$x_1'x_2x_3$	1	0	0	0
1	0	0	0		0	0	0	0
1	0	1	1	$x_1x_2'x_3$	0	1	0	0
1	1	0	1	$x_1x_2x_3'$	0	0	1	0
1	1	1	1	$x_1x_2x_3$	0	0	0	1

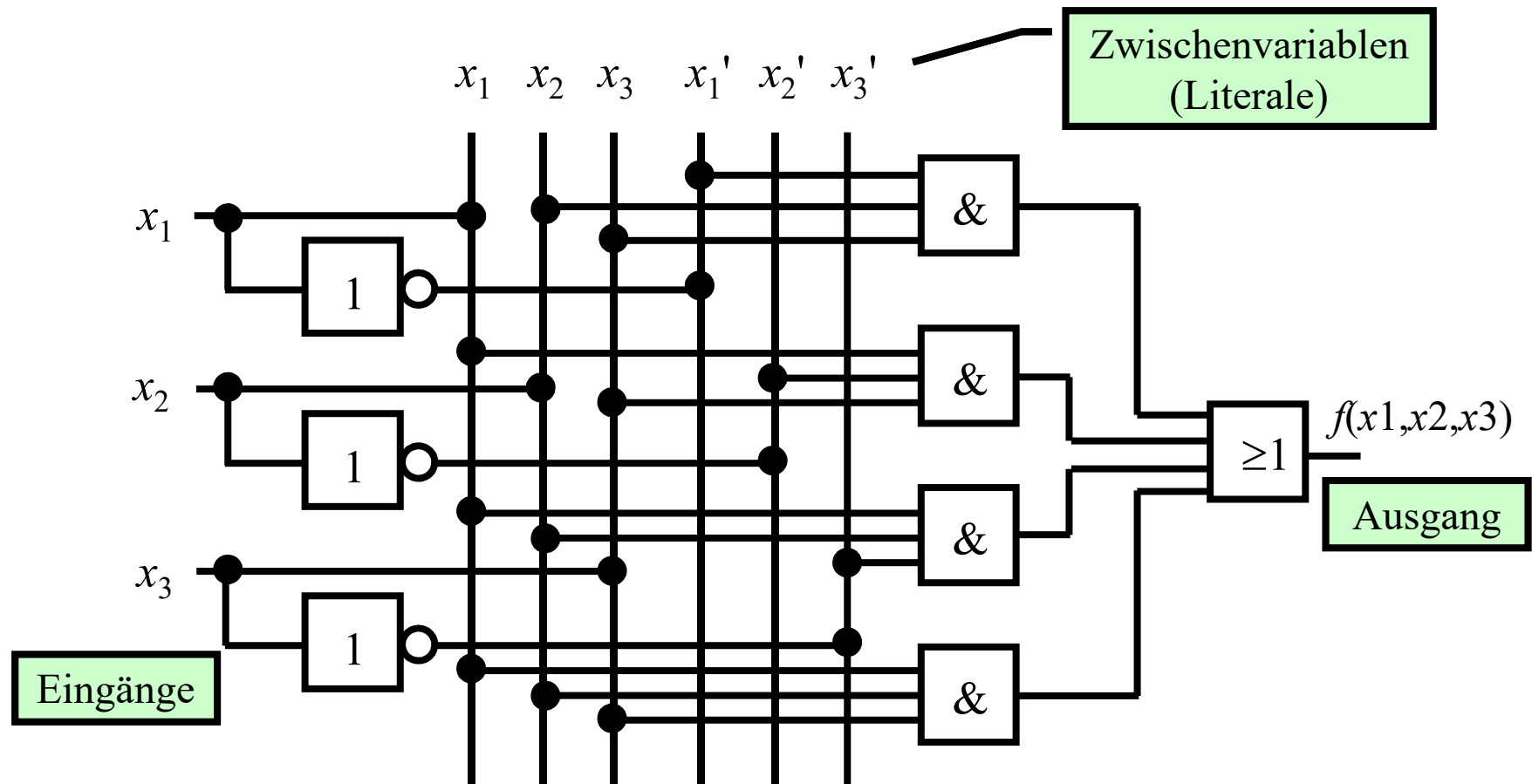
also:

$$f(x_1, x_2, x_3) = x_1'x_2x_3 + x_1x_2'x_3 + x_1x_2x_3' + x_1x_2x_3$$

Beispiel Majorität (2)

- Schaltnetz 1

$$f(x_1, x_2, x_3) = x_1'x_2x_3 + x_1x_2'x_3 + x_1x_2x_3' + x_1x_2x_3$$



Beispiel Majorität (3)

- Vereinfachen

- Aufgabe der Normalform, aber weniger Gatter
- z.B.

$$f(x_1, x_2, x_3) = x_1'x_2x_3 + x_1x_2'x_3 + x_1x_2x_3' + x_1x_2x_3$$

x_2x_3 ausklammern

x_1 ausklammern

$$f(x_1, x_2, x_3) = \underbrace{(x_1' + x_1)}_1 x_2x_3 + x_1(x_2'x_3 + x_2x_3')$$

$$= x_2x_3 + x_1(x_2'x_3 + x_2x_3')$$

Beispiel Majorität (4)

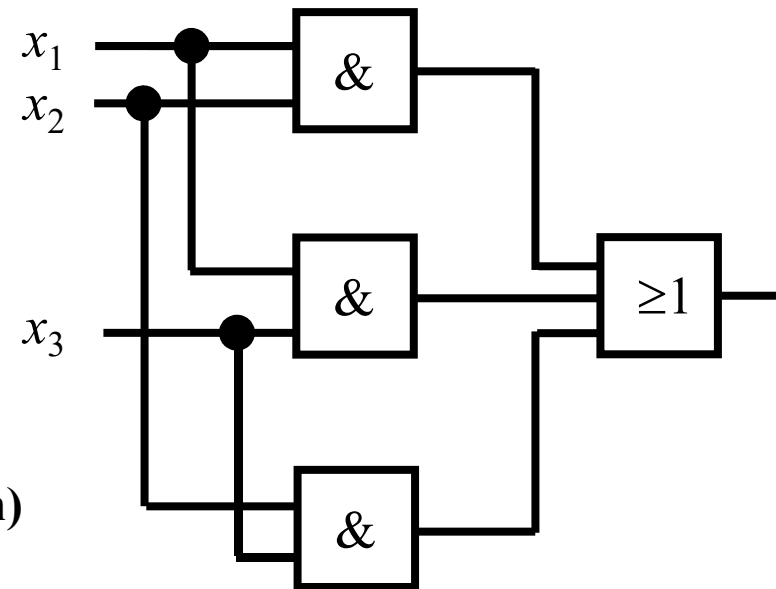
– besser:

zweifaches Kopieren (Idempotenz) von $x_1x_2x_3$:

$$\begin{aligned} f(x_1, x_2, x_3) &= \underbrace{x_1'x_2x_3 + x_1x_2x_3}_{x_2x_3} + \underbrace{x_1x_2'x_3 + x_1x_2x_3}_{x_1x_3} + \underbrace{x_1x_2x_3' + x_1x_2x_3}_{x_1x_2} \\ &= x_2x_3 + x_1x_3 + x_1x_2 \end{aligned}$$

- **Schaltnetz 2**

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$$

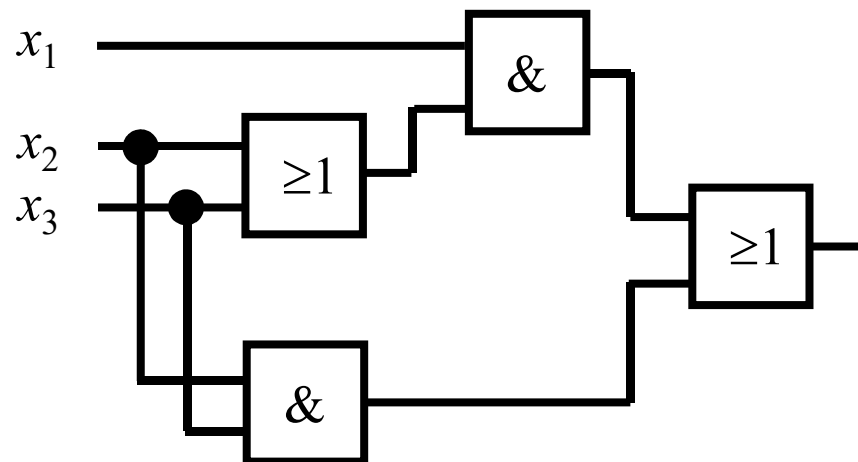


Dies ist keine DNF mehr!
hat aber noch dieselbe Struktur (Polynom)

Beispiel Majorität (5)

- **Schaltnetz 3**

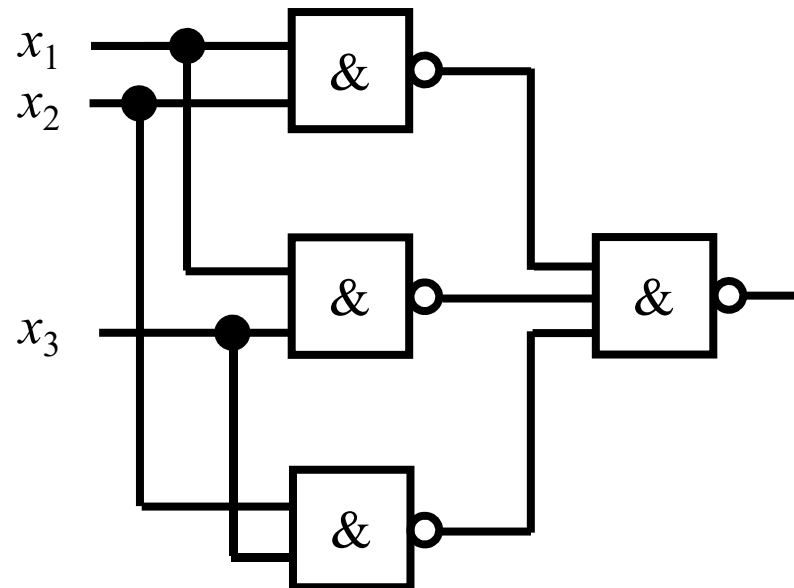
$$\begin{aligned}f(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_2x_3 \\ &= x_1(x_2 + x_3) + x_2x_3\end{aligned}$$



Beispiel Majorität (6)

- Schaltnetz 4

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_2x_3 \\ &= ((x_1x_2)' (x_1x_3)' (x_2x_3)')' \quad (\text{De-Morgan}) \end{aligned}$$



Optimierung von Schaltnetzen

- **Problem**
 - Eine Schaltfunktion, viele mögliche Schaltnetze
 - Welches ist das beste Schaltnetz?
- **Ziele für eine Optimierung**
 - wenige Gatter und wenige Eingänge \Rightarrow billige Hardware
 - Schaltnetz 3 optimal
 - kurze Durchlaufzeit \Rightarrow schnelle Hardware
 - Schaltnetz 2 oder 4 optimal (in CMOS ist 4 optimal, s.u.)
 - Benutzung vorgegebener Gattertypen
 - z.B. nur NAND, NOR, NOT, dann Schaltnetz 4 optimal

Resolutionsregel

- **Resolutionsregel:**

- Wenn in einer disjunktiven Form zwei Monome vorkommen, die sich in genau einer komplementären Variable unterscheiden, so kann man die Monome durch ihren gemeinsamen Teil ersetzen.

- **Beispiel**

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1'x_2x_3 + x_1x_2x_3 \\ &= (x_1' + x_1) x_2x_3 && \text{(Distributivgesetz)} \\ &= 1 x_2x_3 && \text{(Inverses Element)} \\ &= x_2x_3 && \text{(Neutrales Element)} \end{aligned}$$

Resolutionsregel (2)

- **Beachte**

- wegen der Idempotenz $a+a = a$ darf die Resolutionsregel auch mehrfach mit demselben Monom angewandt werden

- **Beispiel**

$$f(x_1, x_2, x_3)$$

$$= x_1'x_2x_3 + x_1x_2'x_3 + x_1x_2x_3$$

$$= (x_1'x_2x_3 + x_1x_2x_3) + (x_1x_2'x_3 + x_1x_2x_3)$$

(Idempotenz)

$$= x_2x_3 + x_1x_3$$

(Resolutionsregel)

KV-Diagramm

- **Verfahren von Karnaugh und Veitch**
 - (Englisch: *Karnaugh map*)
 - grafisches Verfahren zur Vereinfachung von Schaltnetzen
 - sinnvoll bei 3 oder 4 Eingängen
 - hilft beim Erkennen von möglichen Resolutionen
 - Darstellung der Wahrheitstabelle in besonderer Gestalt

		x_1x_2			
		00	01	11	10
x_3	0				
	1				

		x_1x_2			
		00	01	11	10
x_3x_4	00				
	01				
	11				
	10				

KV-Diagramm (2)

- **KV-Diagramm**

- jede Ergebnis-1 aus der Wahrheitstabelle wird an die passende Stelle im KV-Diagramm eingetragen
- **beachte:** beim Übergang von einem Feld zu einem benachbarten Feld ändert sich genau ein Bit
 - siehe Gray-Code
 - damit liegen für die Resolutionsregel geeignete Minterme nebeneinander
- gilt auch zyklisch über die Ränder hinweg

		x_1x_2			
		00	01	11	10
x_3x_4	00				
	01				
	11				
	10				

Beispiel Majorität

- Majoritätsfunktion

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	Minterm
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$x_1'x_2x_3$
1	0	0	0	
1	0	1	1	$x_1x_2'x_3$
1	1	0	1	$x_1x_2x_3'$
1	1	1	1	$x_1x_2x_3$

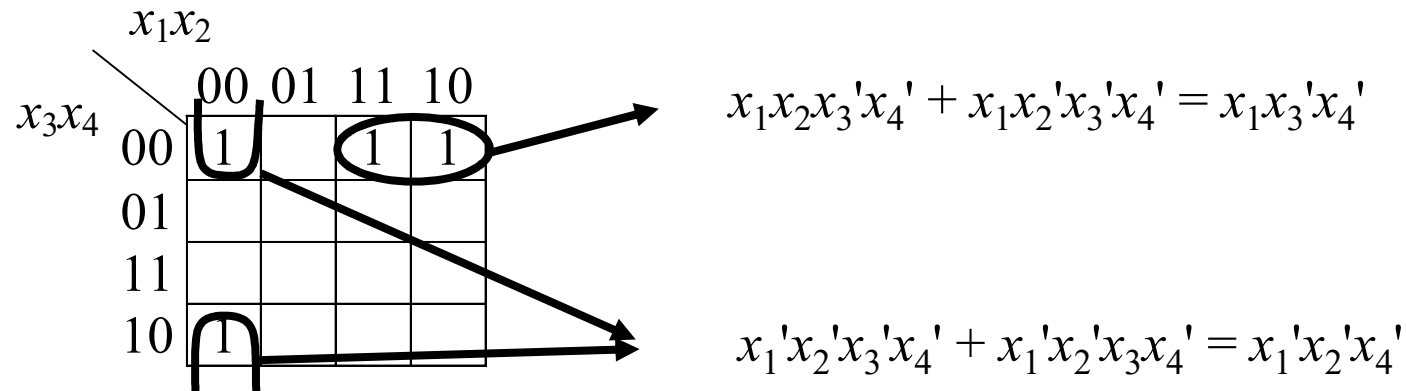
Karnaugh map for the majority function $f(x_1, x_2, x_3)$. The map is a 2x4 grid with x_1x_2 as columns (00, 01, 11, 10) and x_3 as rows (0, 1). The values are: (0,0)=0, (0,1)=0, (1,0)=0, (1,1)=1, (1,0)=1, (1,1)=1, (1,0)=1, (1,1)=1. Arrows connect the minterms from the truth table to their corresponding cells in the K-map: $x_1'x_2x_3$ to (0,1), $x_1x_2'x_3$ to (1,0), $x_1x_2x_3'$ to (1,1), and $x_1x_2x_3$ to (1,1).

Jeder Minterm liefert genau eine 1 im KV-Diagramm

KV-Diagramm (3)

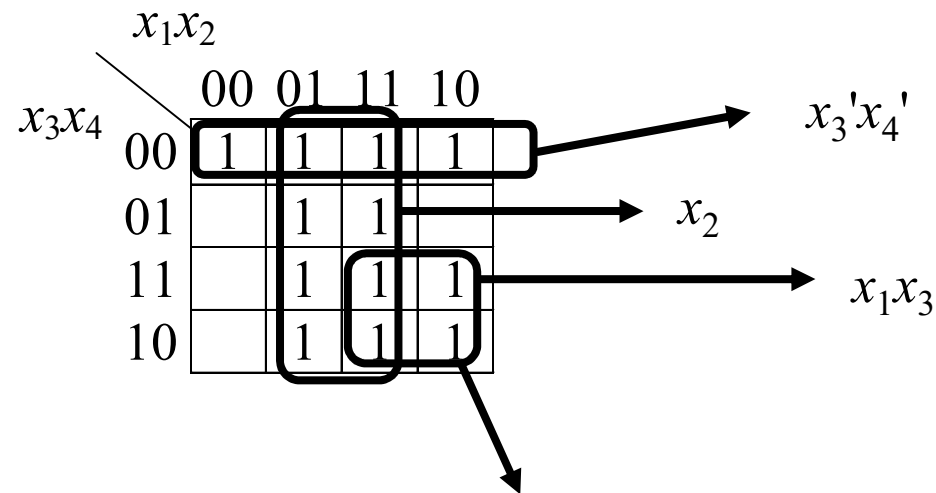
- **Resolutionsblöcke**

- für im KV-Diagramm benachbarte Minterme kann die Resolutionsregel angewandt werden



KV-Diagramm (4)

- die benötigten Monome ergeben sich direkt aus der Randbeschriftung
 - Literale, die sowohl negiert als auch nicht negiert auftreten, fallen weg
 - das Monom besteht aus den restlichen Literalen in der durch die Randbeschriftung festgelegten Ausprägung
- das gilt auch für größere Blöcke der Kantenlänge $2^n * 2^m$
 - sie entstehen durch Verschmelzen kleinerer Resolutionsblöcke
 - die größtmöglichen Blöcke werden auch *Primimplikanten* genannt

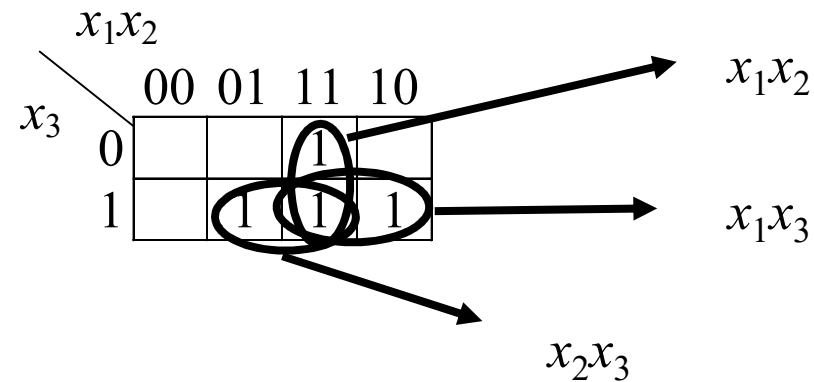


z.B.:

$$\begin{aligned}
 &x_1x_2x_3x_4 + x_1x_2'x_3x_4 + x_1x_2x_3x_4' + x_1x_2'x_3x_4' = \\
 &\quad x_1x_3x_4 \quad + \quad x_1x_3x_4' = x_1x_3
 \end{aligned}$$

Beispiel Majorität (2)

- Beispiel Majorität



– also

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$$

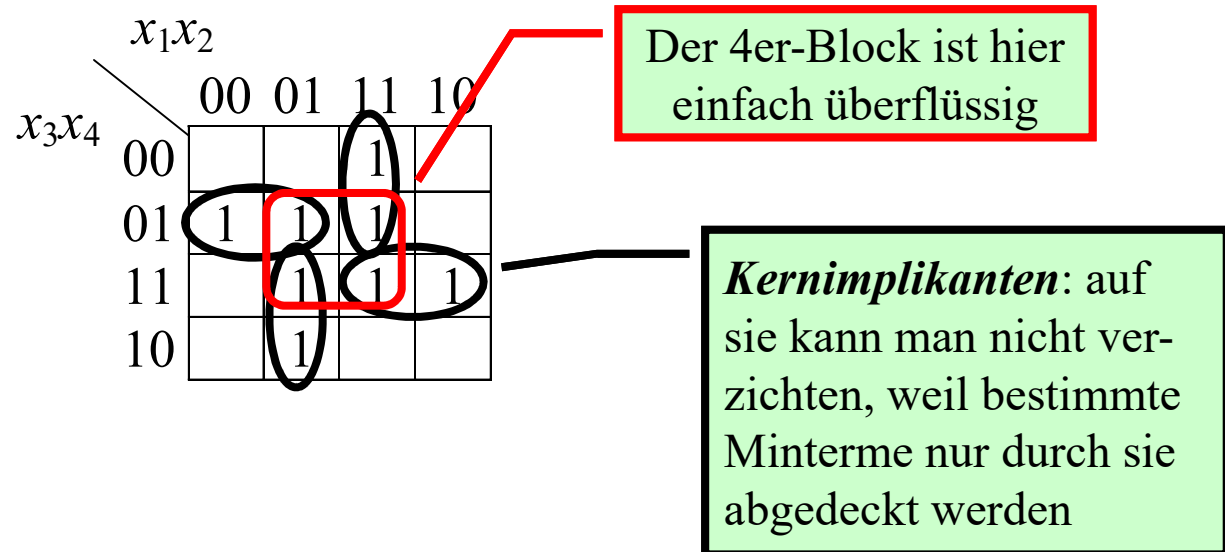
KV-Diagramm (5)

- **Wahl der Resolutionsblöcke**
 - alle Einsen abdecken
 - möglichst große Blöcke wählen
 - Monome enthalten weniger Literale (kleinere Monome)
 - da viele Einsen abgedeckt werden, kommt man in der Regel mit weniger Blöcken aus (weniger Monome)
 - Blöcke dürfen überlappen
 - Ausnutzen der Idempotenz

KV-Diagramm (6)

- **Achtung**

- beachte: es ist nicht immer sinnvoll, den größten Block zu verwenden
- Beispiel:



4 Monome mit je 3 Literalen reichen aus

Don't-Cares

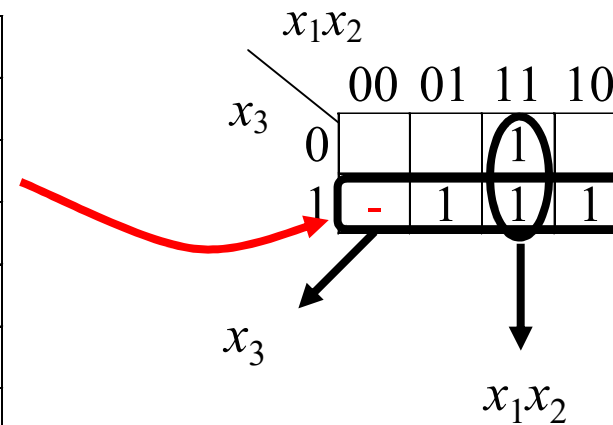
- **Don't-Cares**

- Häufig ist die Ausgabe für bestimmte Eingabekombinationen nicht definiert, also beliebig
- dies kann zur weiteren Vereinfachung der Schaltfunktionen ausgenutzt werden
- Beispiel:
 - angenommen, wir wissen, dass es bei der Majorität nie vorkommt, dass $x_1=x_2=0$ und $x_3=1$ sind (sinnvolleres Beispiel in den Übungen), dann interessiert uns die Ausgabe für diesen Fall nicht

Don't-Cares (2)

- Beispiel Majorität

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	Minterm
0	0	0	0	
0	0	1	-	$x_1'x_2'x_3$
0	1	0	0	
0	1	1	1	$x_1'x_2x_3$
1	0	0	0	
1	0	1	1	$x_1x_2'x_3$
1	1	0	1	$x_1x_2x_3'$
1	1	1	1	$x_1x_2x_3$



$$f(x_1, x_2, x_3) = x_1x_2 + x_3$$

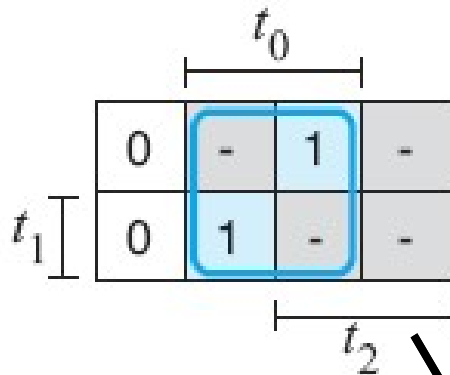
Don't-Care-Term kann 0 oder 1 sein, wie es uns besser passt.
Hier ist 1 sinnvoll.

Don't-Cares (3)

- **Nutzung der Don't-Cares**
 - Annahme: Don't-Care = 1
 - man kann evtl. größere Blöcke wählen, wenn man die Don't-Cares mit überdeckt
 - hilft, Monome mit weniger Literalen zu erhalten
 - Annahme: Don't-Care = 0
 - Don't-Cares brauchen nicht überdeckt zu werden
 - das hilft, Monome zu sparen

KV-Diagramm (7)

- Alternative Darstellung im Lehrbuch



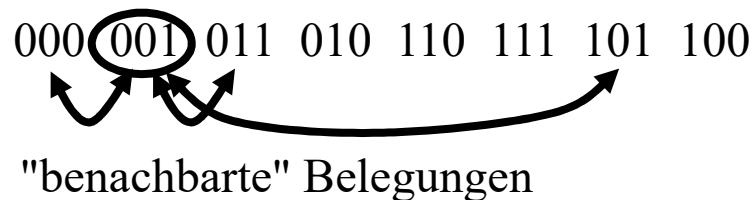
Spalten (oder Zeilen) in denen die angegebene Variable den Wert 1 hat

		$t_2 t_0$			
		00	01	11	10
t_1	0		-	1	-
	1		1	-	-

Bevorzugte Darstellung: leichter und schneller auszufüllen (meine subjektive Meinung)

KV-Diagramm (8)

- **Abschließende Bemerkungen**
 - Schaltfunktionen in KNF
 - können auch mit KV-Diagramm minimiert werden
 - dann fasst man entsprechend die Terme mit Nullen zusammen
 - Anzahl der Schaltvariablen
 - KV-Diagramm bis 4 Variablen praktikabel
 - bei mehr als 4 Variablen wird das KV-Diagramm zu unübersichtlich
 - Gray-Code mit 3 Variablen



- Alternative: 3-dimensionale Darstellung
 - » auf dem Papier auch schwierig
- systematisches Verfahren wird benötigt

Quine-McCluskey Verfahren

- systematisches Verfahren, das mit Tabellen arbeitet
- kann Schaltfunktionen mit vielen Variablen minimieren
- leicht automatisierbar
- benutzt DNF als Ausgangspunkt
- findet systematisch alle Minterme, die nach der Resolutionsregel zusammengefasst werden können

Quine-McCluskey (2)

- **Schreibweise**

- Monome der Schaltfunktion werden durch ihr Binäräquivalent dargestellt
 - "1" steht für nicht negierte Variable
 - "0" für negierte Variable
 - "-" für nicht auftretende Variable

- **Beispiele**

$$x_4 \bar{x}_3 x_2 \bar{x}_1 \quad 1010$$

$$x_4 \bar{x}_3 \bar{x}_1 \quad 10-0$$

$$x_4 x_2 \quad 1-1-$$

Quine-McCluskey (3)

- **Erläuterung des Verfahrens an Beispiel**

- Schaltfunktion sei durch ihre Wertetabelle gegeben
- die Reihenfolge der Funktionswerte wird so gewählt, dass die Binäräquivalente der Minterme aufsteigenden Binärzahlen entsprechen
- als weitere Vereinfachung verwendet man Dezimalzahlen als Indizes für die Minterme

Dez	x_4	x_3	x_2	x_1	$f(x_4, x_3, x_2, x_1)$
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Quine-McCluskey (4)

- **1. Schritt:**

- die Minterme werden in gewichteten Gruppen zusammengefasst
- das Gewicht einer Gruppe ist die *Anzahl* der Einsen in den Binäräquivalenten
 - Gruppe 0: keine 1
 - Gruppe 1: eine 1
 - Gruppe 2: zwei 1'en
 - etc.
- das spart Zeit beim Suchen der passenden Monome für die Resolutionsregel
 - passende Monome liegen in benachbarten Gruppen

Dez	x_4	x_3	x_2	x_1		Gruppe
0	0	0	0	0	✓	0
2	0	0	1	0	✓	1
4	0	1	0	0	✓	
5	0	1	0	1	✓	2
6	0	1	1	0	✓	
10	1	0	1	0	✓	
7	0	1	1	1	✓	3
11	1	0	1	1	✓	

Quine-McCluskey (5)

- **2. Schritt**

- entsprechend der Resolutionsregel werden Monome aus benachbarten Gruppen zusammengefasst
- das ist möglich, wenn sie sich nur an einer Stelle unterscheiden
- man versucht jedes Monom einer Gruppe mit jedem Monom der nächsten Gruppe zu verschmelzen
- alle Monome, die verschmolzen werden können, werden gekennzeichnet
- es bleiben die nicht gekennzeichneten Monome übrig
 - sie werden *Primimplikanten* genannt
 - sie entsprechen den größtmöglichen Resolutionsblöcken im KV-Diagramm, da sie nicht zu noch größeren Blöcken verschmolzen werden können

Quine-McCluskey (6)

Dez	x_4	x_3	x_2	x_1		Gruppe
0,2	0	0	-	0	✓	0
0,4	0	-	0	0	✓	
2,6	0	-	1	0	✓	1
2,10	-	0	1	0	←	Primimplikant
4,5	0	1	0	-	✓	
4,6	0	1	-	0	✓	
5,7	0	1	-	1	✓	2
6,7	0	1	1	-	✓	
10,11	1	0	1	-	←	Primimplikant

Quine-McCluskey (7)

- die beiden Schritte werden solange wiederholt, bis keine Verschmelzung mehr möglich ist
- dabei werden mehrfach entstehende Monome bis auf einen gestrichen
- die Schaltfunktion setzt sich nun nur noch aus den Primimplikanten zusammen

Dez	x_4	x_3	x_2	x_1	Gruppe
0,2 ; 4,6	0	-	-	0	0
0,4 ; 2,6	0	-	-	0	← Primimplikant
4,5 ; 6,7	0	1	-	-	1
4,6 ; 5,7	0	1	-	-	← Primimplikant

Quine-McCluskey (8)

- für Beispiel gilt

$$f(x_4, x_3, x_2, x_1) = (2, 10) + (10, 11) + (0, 2, 4, 6) + (4, 5, 6, 7)$$

- oder in Boolescher Form

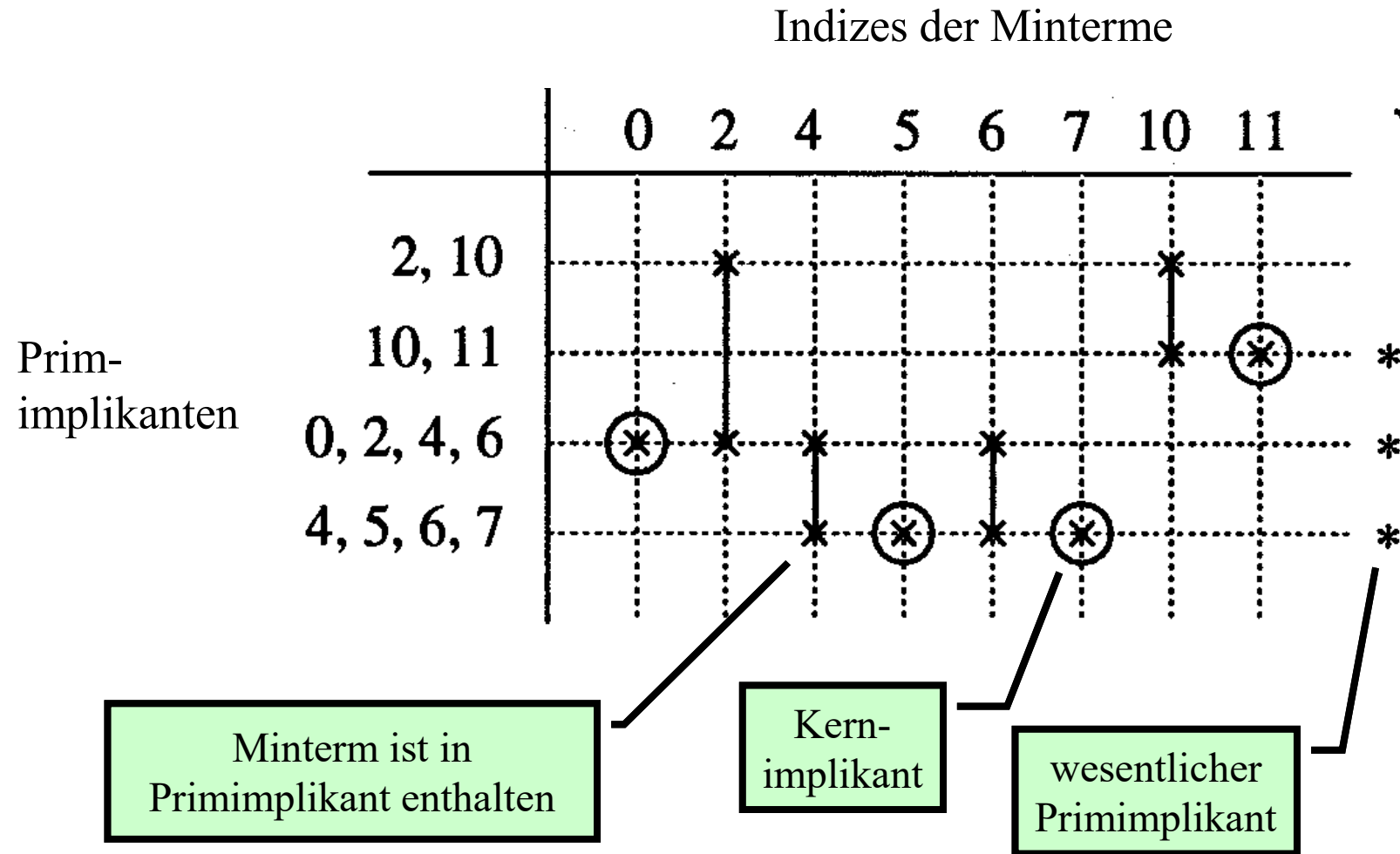
$$f(x_4, x_3, x_2, x_1) = \bar{x}_3 x_2 \bar{x}_1 + x_4 \bar{x}_3 x_2 + \bar{x}_4 \bar{x}_1 + \bar{x}_4 x_3$$

Quine-McCluskey (9)

- **3. Schritt**

- diese Schaltfunktion lässt sich mithilfe von Primimplikantentafeln weiter vereinfachen
- jeder Primimplikant ist aus bestimmten Mintermen entstanden
- andererseits kann jeder Minterm in verschiedenen Primimplikanten enthalten sein
- das Ziel ist es, eine minimale Anzahl von Primimplikanten zu finden, die alle Minterme überdecken
 - diese nennt man *wesentliche Primimplikanten*
 - entspricht den Resolutionsblöcken beim KV-Diagramm, die letztendlich benutzt werden

Quine-McCluskey (10)



Quine-McCluskey (11)

- Steht in einer Spalte nur ein Primimplikant, so nennt man ihn
 - *Kernimplikant*
- er muss in der Minimalform erscheinen, da nur er den Minterm abdeckt, und wird deshalb mit einem Kreis markiert
- die Minterme, die dieser wesentliche Primimplikant überdeckt, werden durch "|" verbunden und damit gestrichen
- aus den evtl. verbleibenden Primimplikanten sucht man eine minimale Anzahl heraus, die alle verbleibenden Minterme überdecken
 - *minimale Restüberdeckung*
 - muss nicht unbedingt eindeutig sein
- die minimierte Schaltfunktion ist die Disjunktion (ODER-Verknüpfung) der Kernimplikanten und der Restüberdeckung

$$f(x_4, x_3, x_2, x_1) = x_4 \bar{x}_3 x_2 + \bar{x}_4 \bar{x}_1 + \bar{x}_4 x_3$$

Quine-McCluskey und Don't-Cares

- **Erweiterung des Verfahrens zur Berücksichtigung von Don't-Care Belegungen**
 - zunächst werden alle don't-care Belegungen wie 1'en behandelt
 - dadurch entstehen mehr Möglichkeiten Belegungen zusammenzufassen
 - die Chance für größere Primimplikanten steigt
 - in der Primimplikantentafel
 - müssen nur diejenigen Minterme abgedeckt werden, die tatsächlich eine 1 erfordern
 - die Don't-Care Belegungen können, müssen aber nicht abgedeckt werden
 - im nachfolgenden Beispiel aus dem Buch werden die Don't-Care Belegungen leider immer als 0 genutzt

Quine-McCluskey und Don't-Cares (2)

Implikanten
nullter Ordnung:

	x_4	x_3	x_2	x_1	
1	0	0	0	1	✓
3	0	0	1	1	✓
5	0	1	0	1	✓
7	0	1	1	1	✓
10	1	0	1	0	✓
11	1	0	1	1	✓
12	1	1	0	0	✓
13	1	1	0	1	✓
14	1	1	1	0	✓
15	1	1	1	1	✓

Implikanten
erster Ordnung:

	x_4	x_3	x_2	x_1	
1,3	0	0	-	1	✓
1,5	0	-	0	1	✓
3,7	0	-	1	1	✓
3,11	-	0	1	1	✓
5,7	0	1	-	1	✓
5,13	-	1	0	1	✓
7,15	-	1	1	1	✓
10,11	1	0	1	-	✓
10,14	1	-	1	0	✓
11,15	1	-	1	1	✓
12,13	1	1	0	-	✓
12,14	1	1	-	0	✓
13,15	1	1	-	1	✓
14,15	1	1	1	-	✓

Implikanten
zweiter Ordnung:

	x_4	x_3	x_2	x_1
1,3,5,7	0	-	-	1
3,7,11,15	-	-	1	1
12,13,14,15	1	1	-	-
5,7,13,15	-	1	-	1
10,11,14,15	1	-	1	-

Don't-Cares

	x_4	x_3	x_2	x_1
1,3,5,7	0	-	-	1
3,7,11,15	-	-	1	1
12,13,14,15	1	1	-	-
5,7,13,15	-	1	-	1
10,11,14,15	1	-	1	-

Don't-Cares

	1	3	5	7	10	11	12	13	14	15
1	×									
3		×								
5			×							
7				×						
10					×	×				
11						×				
12							×	×	×	×
13							×	×		×
14								×	×	×
15									×	×

$\Rightarrow x_1 x_4'$