

DIGITALE SCHALTUNGEN



Robert Wille (robert.wille@jku.at)

Sebastian Pointner (sebastian.pointner@jku.at)

Institut für Integrierte Schaltungen

Abteilung für Schaltkreis- und Systementwurf

INHALT DER VORLESUNG



■ Grundlagen

- ☐ Beschreibungen über „0“ und „1“ (Boolesche Algebra)
- ☐ Beschreibungen von Schaltungen

■ Rechnen

- ☐ Darstellung von Zahlen
- ☐ Digitale Schaltungen für Addition, Subtraktion, Multiplikation

■ Speichern

- ☐ Sequentielle Schaltungen
- ☐ Speicherelemente

■ Steuern

- ☐ Endliche Automaten
- ☐ Synthese von Steuerwerken

■ Entwerfen

- ☐ Synthese von allgemeinen Schaltungen
- ☐ Logikminimierung

INHALT DER VORLESUNG



■ Grundlagen

- ☐ Beschreibungen über „0“ und „1“ (Boolesche Algebra)
- ☐ Beschreibungen von Schaltungen

■ Rechnen

- ☐ Darstellung von Zahlen
- ☐ Digitale Schaltungen für Addition, Subtraktion, Multiplikation

■ Speichern

- ☐ Sequentielle Schaltungen
- ☐ Speicherelemente

■ Steuern

- ☐ Endliche Automaten
- ☐ Synthese von Steuerwerken

■ Entwerfen

- ☐ **Synthese von allgemeinen Schaltungen**
- ☐ **Logikminimierung**

ENTWERFEN



Robert Wille (robert.wille@jku.at)

Sebastian Pointner (sebastian.pointner@jku.at)

Institut für Integrierte Schaltungen

Abteilung für Schaltkreis- und Systementwurf

LOGIKSYNTHESE UND -MINIMIERUNG

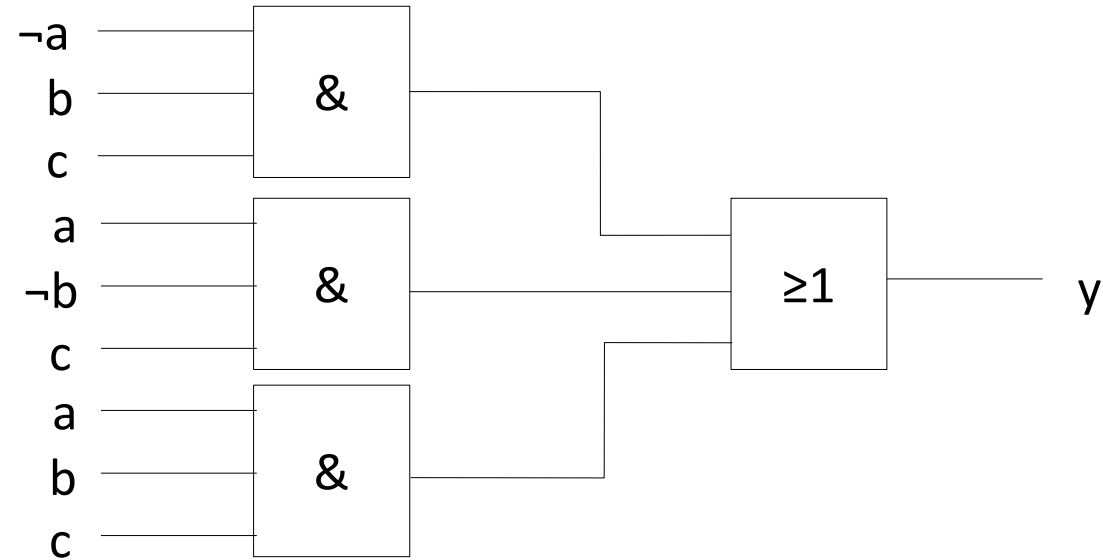
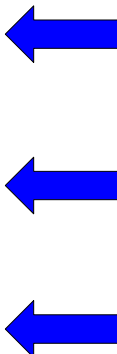
- Dedizierte Schaltungen lassen sich idR nicht gut/effizient „von Hand“ entwickeln
→ Methodik für Schaltkreissynthese
- Logiksynthese und –minimierung: Versuch, Logik der Schaltung
 - ☐ zu generieren bzw.
 - ☐ zu minimieren
- Unterscheidung zwischen
 - ☐ kombinatorischer Synthese
(Realisierung einer Booleschen Funktion)
 - ☐ sequentieller Synthese
(Realisierung eines endlichen Automaten;
kann als Boolesche Funktion repräsentiert werden)
- Ausgangspunkt: Funktionsbeschreibung (Wahrheitstabelle, Boolescher Ausdruck, ...)

WDHLG.: SYNTHESE (SIMPEL)

- Realisierung beliebiger Wahrheitstabellen durch Grundgatter möglich
- Vorgehen:
 - ☐ Für jede Zeile mit Ausgabewert 1:
 - Und-Gatter mit passender Eingangsbeschaltung
 - ☐ Oder-Verknüpfung aller Und-Gatter
- Funktioniert für alle Tabellen, aber
 - ☐ teuer und
 - ☐ nicht skalierbar

WDH.: SYNTHESE (SIMPEL) – BEISPIEL

a	b	c	y
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	0
1	1	1	1



WEITERE GRUNDLAGEN

- Die Booleschen Ausdrücke x_i und x_i' heißen **Literale**, wobei x_i als **positives Literal** und x_i' als **negatives Literal** bezeichnet wird.
- Eine Konjunktion von Literalen wird mit **Monom** bezeichnet, wenn zusätzlich folgendes gilt:
 - ☐ jedes Literal kommt höchstens einmal vor
 - ☐ es kommt nicht sowohl das positive als auch das negative Literal einer Variable vor
- Ein Monom heißt **vollständig** oder **Minterm**, wenn jede Variable entweder als positives oder als negatives Literal vorkommt.
- Eine Disjunktion von paarweise verschiedenen Monomen heißt **Polynom**. Sind alle Monome des Polynoms vollständig, so heißt das Polynom **vollständig**.

BEISPIEL / VERANSCHAULICHUNG DURCH WÜRFEL

- Jede Boolesche Funktion f in n Variablen und einem Ausgang kann über einen n -dimensionalen Würfel durch Markierung der $ON(f)$ -Menge spezifiziert werden.

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = & \\ & x_1 x_2 x_3' x_4' + \\ & x_1 x_2 x_3' x_4 + \\ & x_1 x_2 x_3 x_4' + \\ & x_1 x_2 x_3 x_4 + \end{aligned}$$

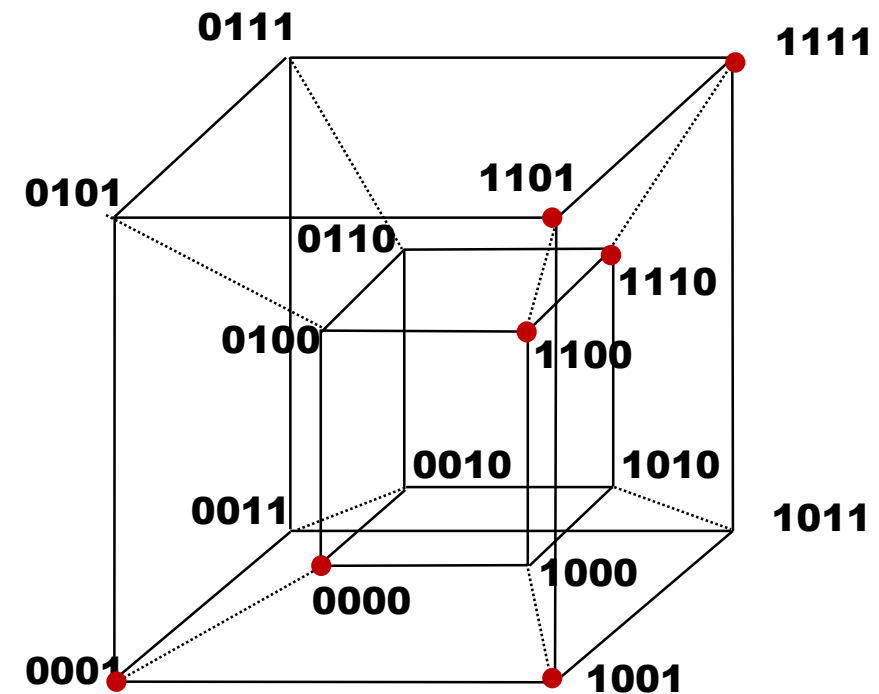
$$\begin{aligned} & x_1' x_2' x_3' x_4' + \\ & x_1' x_2' x_3' x_4 + \end{aligned}$$

$$x_1 x_2' x_3' x_4$$

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = & \\ & x_1 x_2 + \end{aligned}$$

$$x_1' x_2' x_3' +$$

$$x_1 x_2' x_3' x_4$$



BEISPIEL / VERANSCHAULICHUNG DURCH WÜRFEL

- Jede Boolesche Funktion f in n Variablen und einem Ausgang kann über einen n -dimensionalen Würfel durch Markierung der $ON(f)$ -Menge spezifiziert werden.

$$f(x_1, x_2, x_3, x_4) =$$
$$x_1 x_2 x_3' x_4' +$$
$$x_1 x_2 x_3' x_4 +$$
$$x_1 x_2 x_3 x_4' +$$
$$x_1 x_2 x_3 x_4 +$$

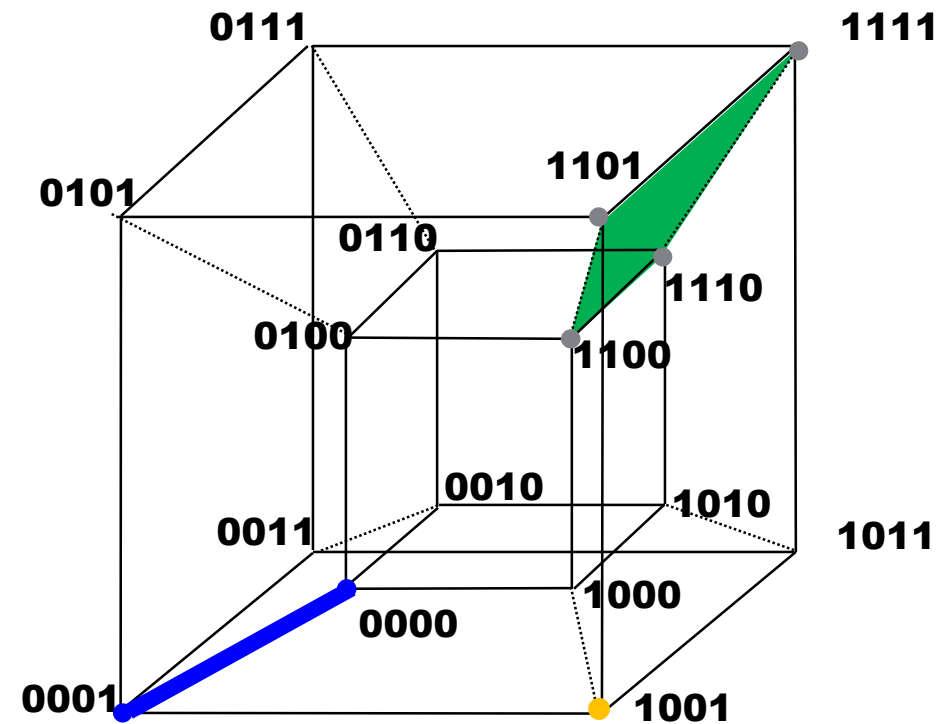
$$x_1' x_2' x_3' x_4' +$$
$$x_1' x_2' x_3' x_4 +$$

$$x_1 x_2' x_3' x_4$$

$$f(x_1, x_2, x_3, x_4) =$$
$$x_1 x_2 +$$

$$x_1' x_2' x_3' +$$

$$x_1 x_2' x_3' x_4$$



LOGIKMINIMIERUNG

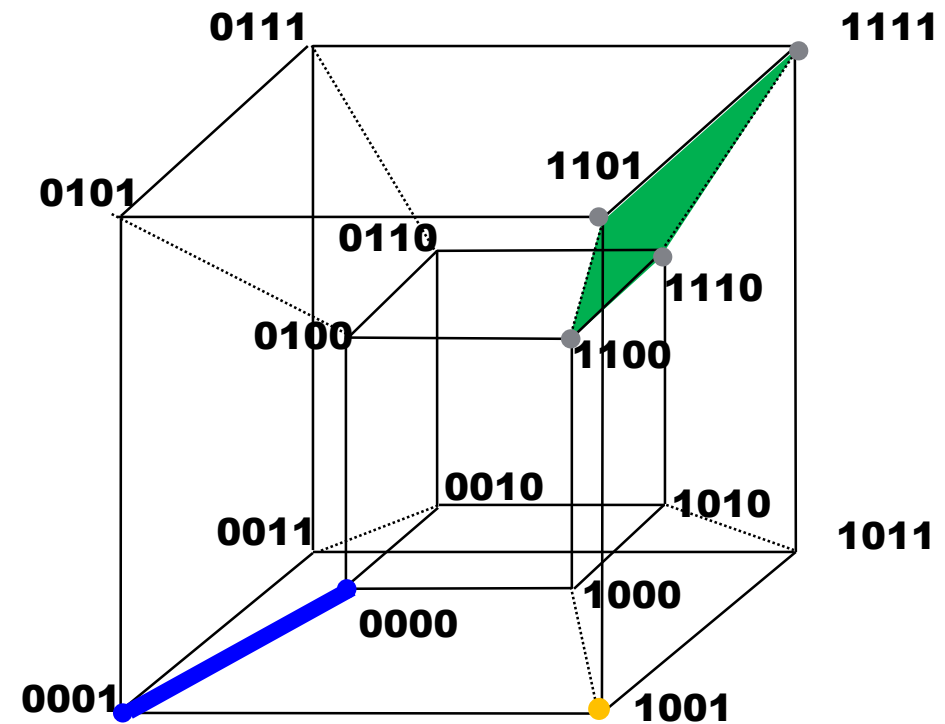
■ Gegeben

Eine Boolesche Funktion f in n Variablen und einem Ausgang in Form eines markierten n -dimensionalen Würfels

■ Gesucht

Eine minimale Überdeckung der markierten Knoten durch maximale Teilwürfel im n -dimensionalen Würfel.

□ Minimal: mit minimal vielen Teilwürfeln



NOCH MEHR GRUNDLAGEN....

■ Sei f eine Boolesche Funktion mit einem Ausgang.

□ Ein **Implikant von f** ist ein Monom q mit $\psi(q) \leq f$.

Ein Monom m ist genau dann ein Implikant von f , wenn entweder

● m ein Minterm von f ist, oder

● $m \cdot x$ und $m \cdot x'$ Implikanten von f sind für eine Variable x , die nicht in m vorkommt

□ Ein **Primimplikant von f** ist ein maximaler Implikant q von f , d.h. es gibt keinen Implikanten s ($s \neq q$) von f mit $\psi(q) \leq \psi(s)$.

→ Die Monome eines Polynoms p von f sind alle Implikanten von f .

■ Ein **Minimalpolynom p** einer Booleschen Funktion f ist ein Polynom von f mit minimalen Kosten, d.h. mit der Eigenschaft $\text{cost}(p) \leq \text{cost}(p')$ für jedes Polynom p' von f .

→ Jedes Minimalpolynom p einer Booleschen Funktion f besteht ausschließlich aus Primimplikanten von f .

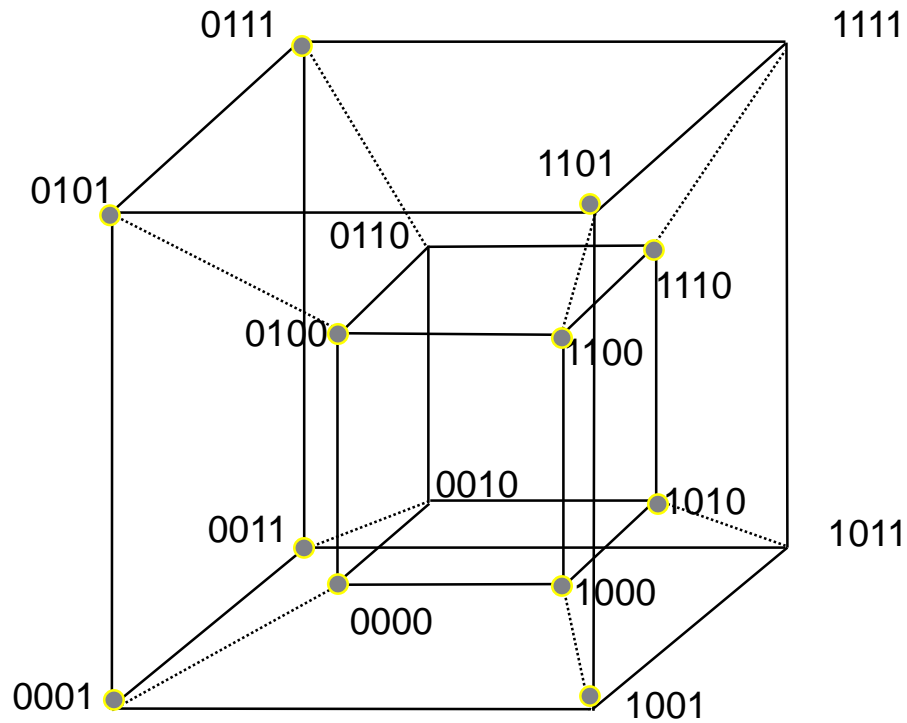
EINSCHUB: KOSTEN

- Die **primären Kosten** $\text{cost}_1(p)$ sind gleich der Anzahl der Monome in p
- Die **sekundären Kosten** $\text{cost}_2(p)$ sind gleich der Anzahl der Literale in p plus die Anzahl der Monome in p .
- Sei im Folgenden **cost** = $(\text{cost}_1, \text{cost}_2)$ die Kostenfunktion mit der Eigenschaft, dass für zwei Polynome p und p' die Ungleichung $\text{cost}(p) \leq \text{cost}(p')$ genau dann gilt, wenn entweder
 - $\text{cost}_1(p) \leq \text{cost}_1(p')$ oder
 - $\text{cost}_1(p) = \text{cost}_1(p')$ und $\text{cost}_2(p) \leq \text{cost}_2(p')$gilt.

OFFEN

- Sei f eine Boolesche Funktion mit einem Ausgang.
 - ☐ Wie berechnet man alle Primimplikanten?
 - ☐ Welche Primimplikanten bilden mein Minimalpolynom? (Überdeckungsproblem)

NOTATION IM FOLGENDEN



$f\{x_1, x_2, x_3, x_4\}$:

0 0 0 0

0 0 0 1

0 1 0 0

1 0 0 0

0 0 1 1

0 1 0 1

1 0 0 1

1 0 1 0

1 1 0 0

0 1 1 1

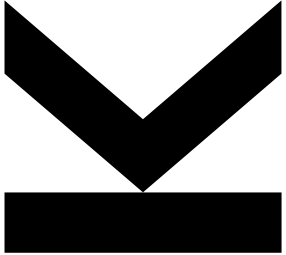
1 1 0 1

1 1 1 0

steht für $x_1'x_2'x_3'x_4'$

steht für $x_1x_2x_3x_4$

K-MAPS



Robert Wille (robert.wille@jku.at)

Sebastian Pointner (sebastian.pointner@jku.at)

Institut für Integrierte Schaltungen

Abteilung für Schaltkreis- und Systementwurf

EINSCHUB GRAY-CODE

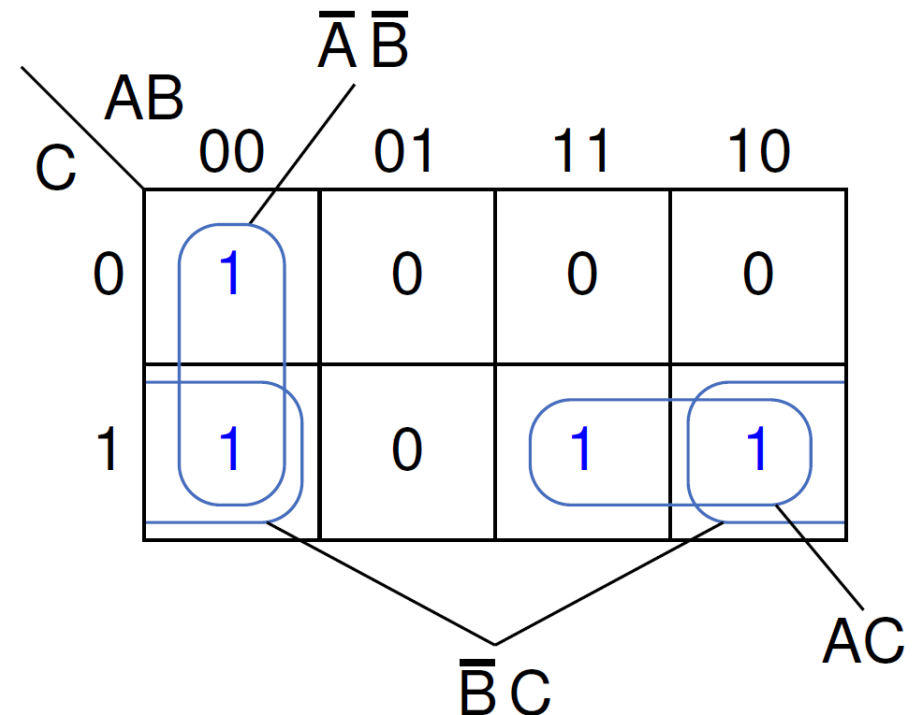
- Bisher Binary-Coded-Decimal (BCD)
 - $1001 = 9$
 - Most-Significant-Bit (MSB) links, Least-Significant-Bit (LSB) rechts
- BCD ist sehr ineffizient:
 - Beispiel Zähler: $0111 \rightarrow 1000$ (von 7 auf 8)
 - Es ändern sich alle 4 Bits
- Hamming Distanz 1
 - Zwei Binäre Zahlen haben genau dann Hamming Distanz 1, wenn sie sich nur in einem Bit unterscheiden.
- Gray-Code: Binär Code mit Hamming Diszanz 1

BCD Zahl	Gray Code Zahl
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

KARNAUGH MAPS

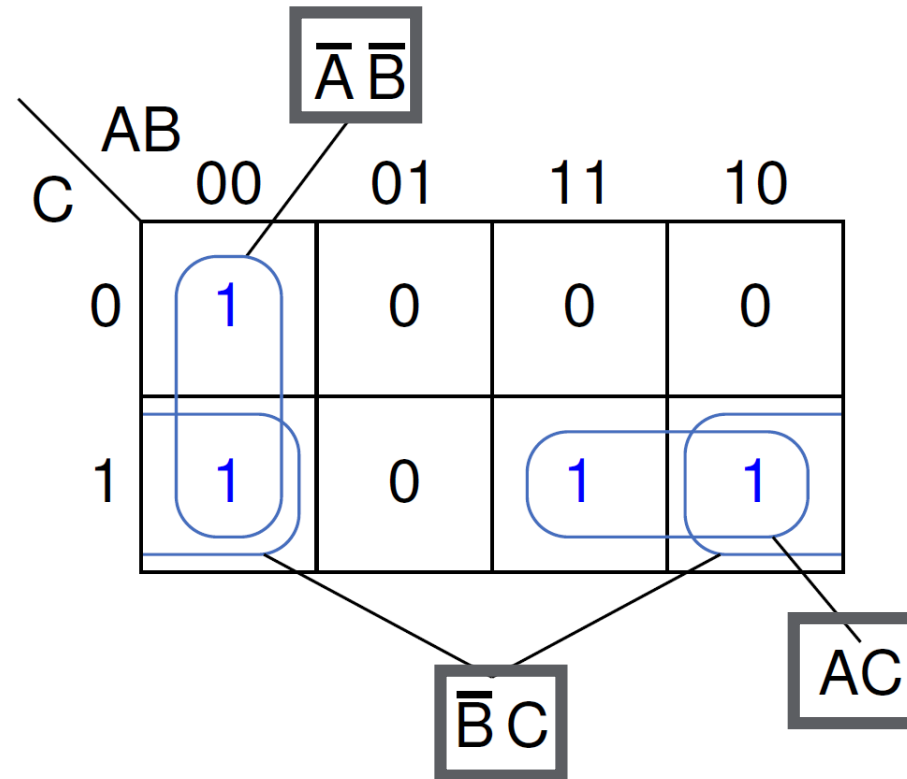
- Darstellung einer Funktion als 2-dimensionales Feld
- Einzelne Zellen geben den Funktionswert des jeweiligen Minterms an
- Benachbarte Zellen unterscheiden sich in ihren Koordinaten nur durch ein Bit (dadurch lassen sich benachbarte Zellen mit gleichem Funktionswert kürzen)

C \ AB	00	01	11	10
	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$



KARNAUGH MAPS

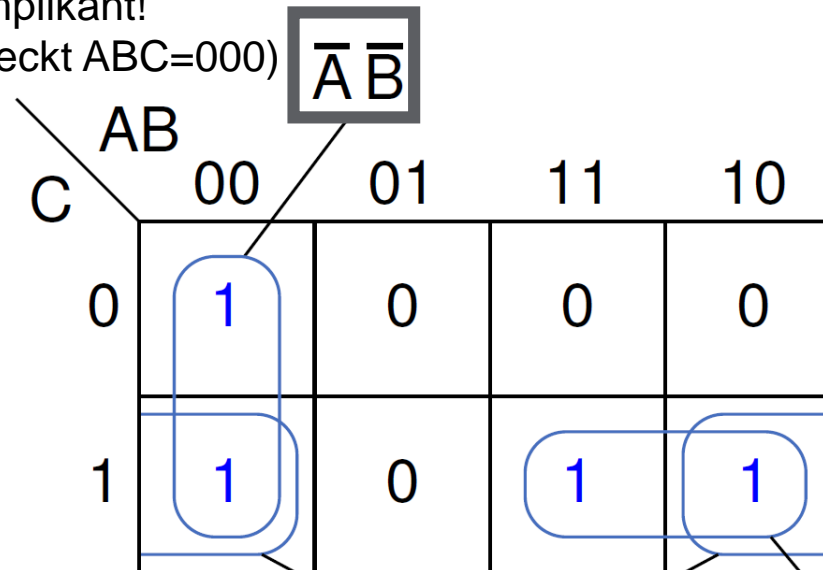
Keine weitere Kürzung möglich!
→ Primimplikanten



KARNAUGH MAPS

Keine weitere Kürzung möglich!
→ Primimplikanten

Wesentlicher Primimplikant!
(kein anderer überdeckt $ABC=000$)



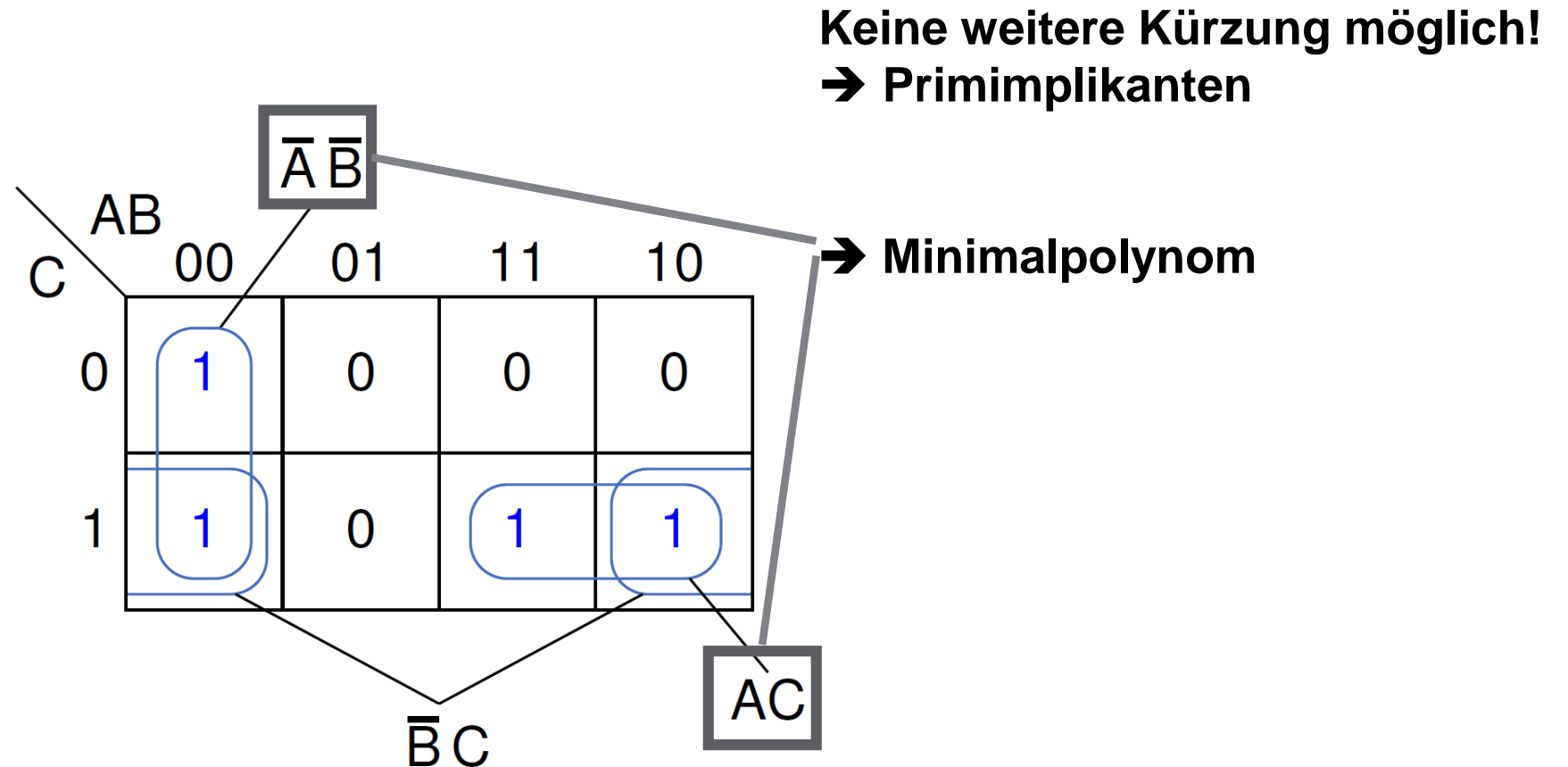
Kann entfallen (wird durch die anderen
Primimplikanten bereits abgedeckt)

$\bar{B}C$

AC

Wesentlicher Primimplikant!
(kein anderer überdeckt $ABC=111$)

KARNAUGH MAPS



LOGIKMINIMIERUNG NACH QUINE/MCCLUSKEY



Robert Wille (robert.wille@jku.at)

Sebastian Pointner (sebastian.pointner@jku.at)

Institut für Integrierte Schaltungen

Abteilung für Schaltkreis- und Systementwurf

GENERIERUNG VON PRIMIMPLIKANTEN NACH QUINE

■ Berechne alle Primimplikanten

polynom function Quine ($f: \mathbf{B}^n \rightarrow \mathbf{B}$)

begin

$L_0 := \text{Minterm}(f);$

$i := 0;$

// L_i enthält alle Implikanten von f der Länge $n-i$

$\text{Prim}(f) := \emptyset;$

while ($L_i \neq \emptyset$) and ($i < n$)

loop $L_{i+1} := \{m \mid mx \text{ und } mx' \text{ sind in } L_i \text{ für ein } x\};$

$\text{Prim}(f) := \text{Prim}(f) \cup$

$\{m \mid m \in L_i \text{ und } m \text{ wird von keinem } q \in L_{i+1} \text{ überdeckt}\};$

$i := i+1;$

pool;

return $\text{Prim}(f) \cup L_i;$

end;

VERBESSERUNG DURCH MCCLUSKEY

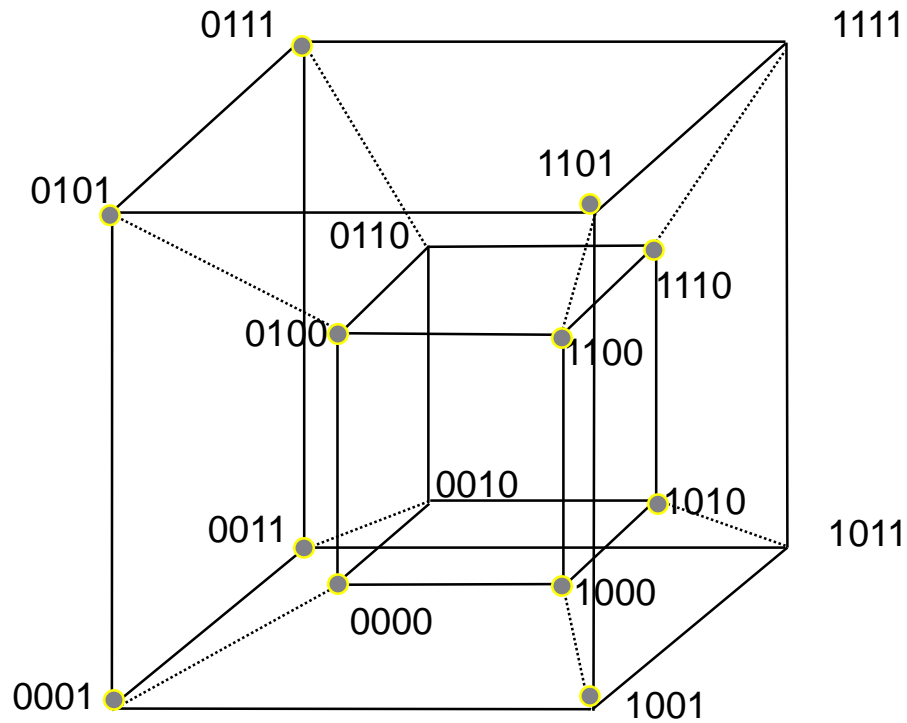
Vergleiche nur Monome untereinander,

- ☐ welche die gleichen Variablen enthalten und
- ☐ bei denen sich die Anzahl der positiven Literale um 1 unterscheidet.

Kann erreicht werden durch

- ☐ Partitioniere L_i in Klassen L_i^M , mit $M \subseteq \{x_1, \dots, x_n\}$ und $|M|=n-i$. L_i^M enthalte die Implikanten aus L_i , deren Literale alle aus M sind.
- ☐ Ordne die Monome in L_i^M gemäß der Anzahl der positiven Literale.

DAS VERFAHREN VON QUINE-MCCLUSKEY: BEISPIEL (1)



$L_0\{x_1, x_2, x_3, x_4\}$:

0 0 0 0

0 0 0 1

0 1 0 0

1 0 0 0

0 0 1 1

0 1 0 1

1 0 0 1

1 0 1 0

1 1 0 0

0 1 1 1

1 1 0 1

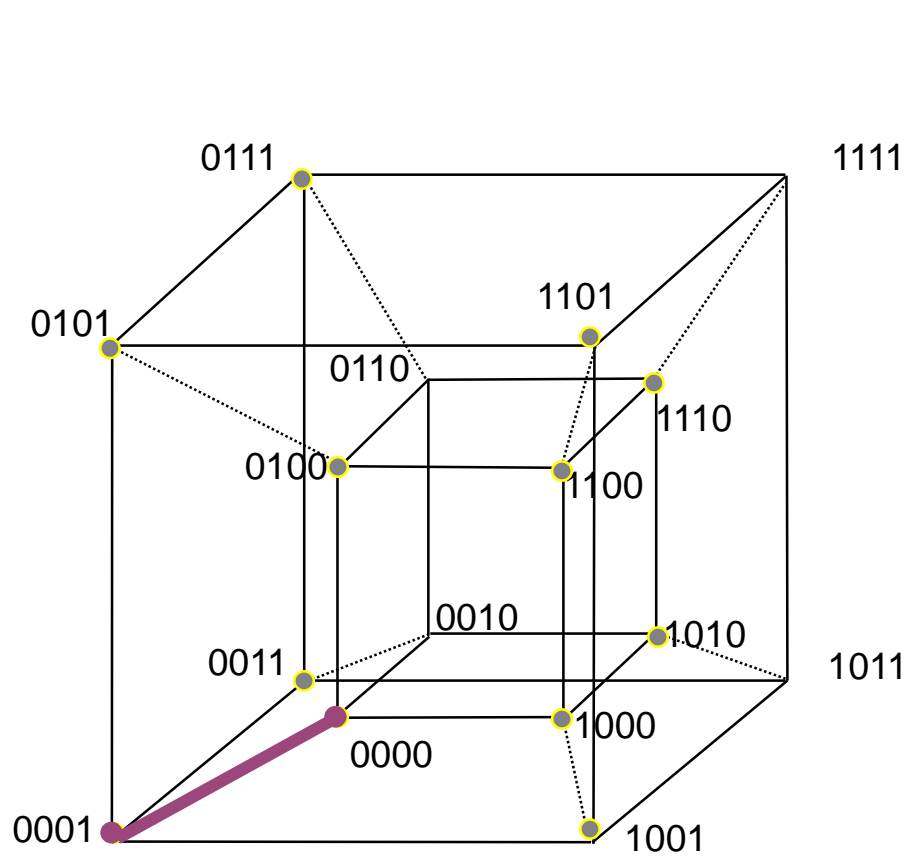
1 1 1 0

steht für $x_1'x_2'x_3'x_4'$

steht für $x_1x_2'x_3'x_4$

Vergleiche im Folgenden nur
Monome aus benachbarten Blöcken!

BEISPIEL QUINE-MCCLUSKEY (2)



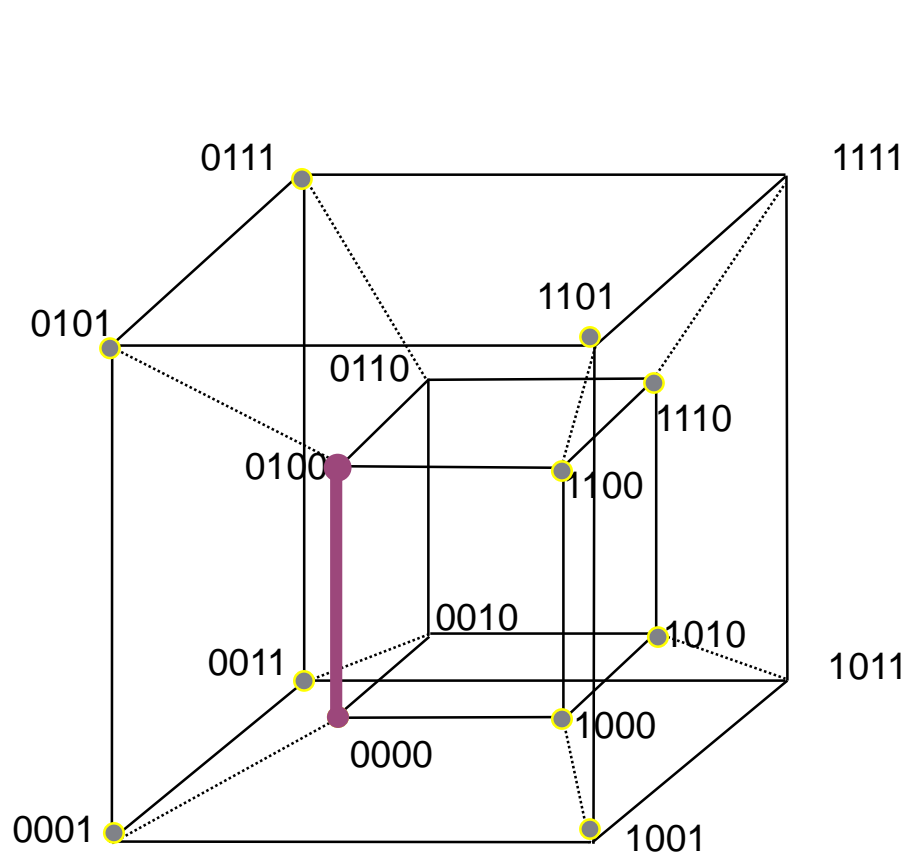
$L_0\{x_1, x_2, x_3, x_4\}$:

\Rightarrow 0 0 0 0
 \Rightarrow 0 0 0 1
 0 1 0 0
1 0 0 0
 0 0 1 1
 0 1 0 1
 1 0 0 1
 1 0 1 0
1 1 0 0
 0 1 1 1
 1 1 0 1
 1 1 1 0

$L_1\{x_1, x_2, x_3\}$:

0 0 0 -

BEISPIEL QUINE-MCCLUSKEY (3)



$L_0\{x_1, x_2, x_3, x_4\}:$

\Rightarrow 0 0 0 0
 0 0 0 1
 \Rightarrow 0 1 0 0
1 0 0 0
 0 0 1 1
 0 1 0 1
 1 0 0 1
 1 0 1 0
1 1 0 0
 0 1 1 1
 1 1 0 1
 1 1 1 0

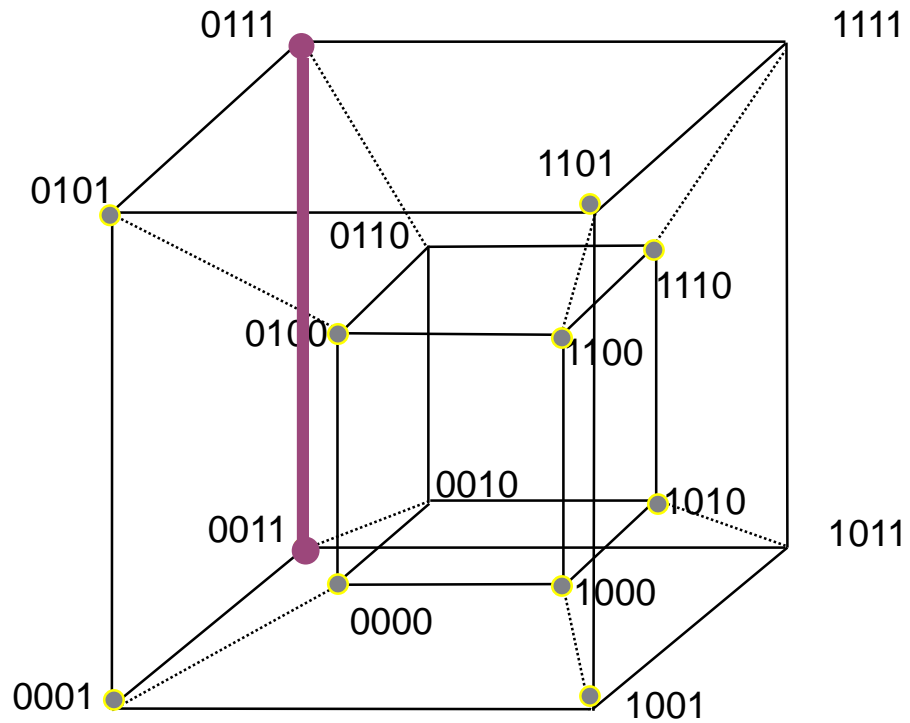
$L_1\{x_1, x_2, x_3\}:$

0 0 0 -

$L_1\{x_1, x_3, x_4\}:$

0 - 0 0

BEISPIEL QUINE-MCCLUSKEY (4)



$L_0\{x_1, x_2, x_3, x_4\}$:

0 0 0 0
 0 0 0 1
 0 1 0 0
1 0 0 0
 → 0 0 1 1
 0 1 0 1
 1 0 0 1
 1 0 1 0
1 1 0 0
 → 0 1 1 1
 1 1 0 1
 1 1 1 0

$L_1\{x_1, x_2, x_3\}$:

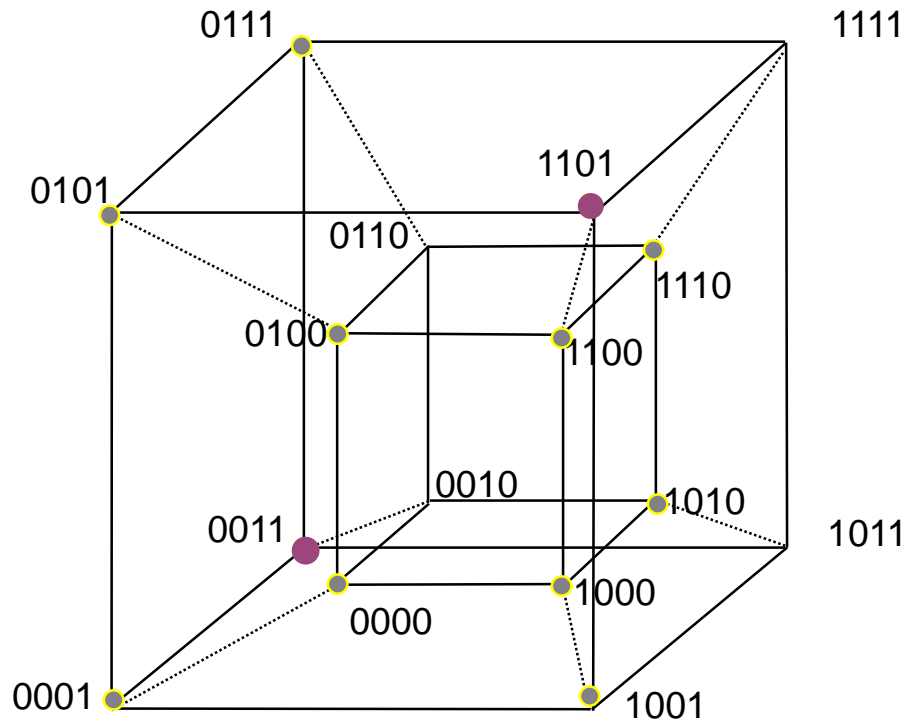
0 0 0 -

$L_1\{x_1, x_3, x_4\}$:

0 - 0 0

.....
0 - 1 1

BEISPIEL QUINE-MCCLUSKEY (5)



$L_0\{x_1, x_2, x_3, x_4\}:$

0 0 0 0
 0 0 0 1
 0 1 0 0
1 0 0 0
 → 0 0 1 1
 0 1 0 1
 1 0 0 1
 1 0 1 0
1 1 0 0
 0 1 1 1
 → 1 1 0 1
 1 1 1 0

$L_1\{x_1, x_2, x_3\}:$

0 0 0 -

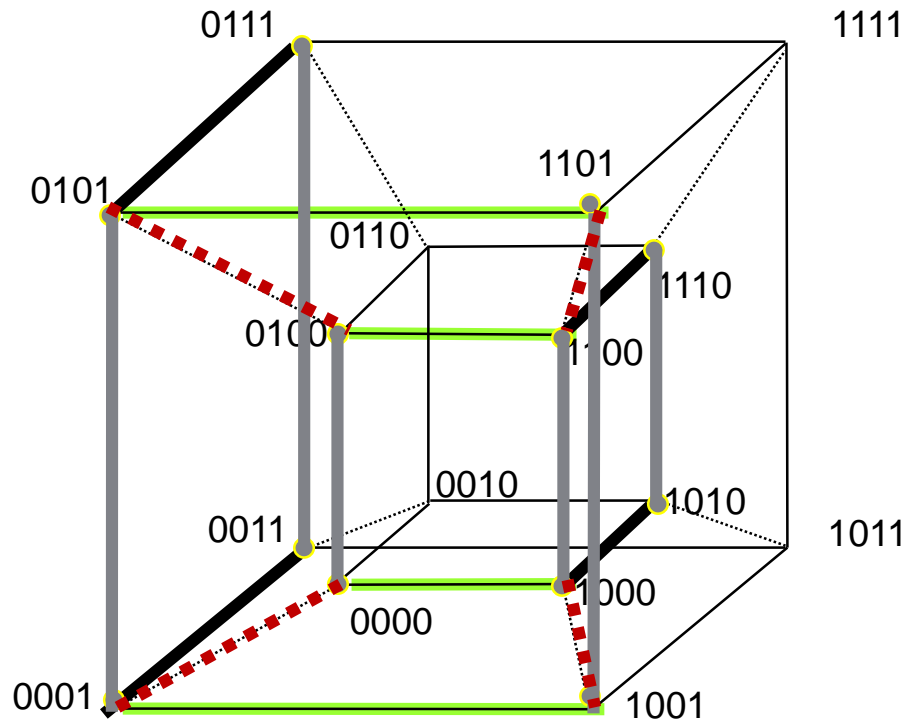
$L_1\{x_1, x_3, x_4\}:$

0 - 0 0

.....
0 - 1 1

**Nicht kürzbar, da nicht
Ecken der gleichen Kante
(Consensus existiert nicht!)**

BEISPIEL QUINE-MCCLUSKEY (6)



$L_1\{x1,x2,x4\}:$

0 0 - 1

1 0 - 0

0 1 - 1

1 1 - 0

$L_1\{x1,x2,x3\}:$

0 0 0 -

0 1 0 -

1 0 0 -

1 1 0 -

$L_1\{x2,x3,x4\}:$

- 0 0 0

- 0 0 1

- 1 0 0

- 1 0 1

$L_1\{x1,x3,x4\}:$

0 - 0 0

0 - 0 1

1 - 0 0

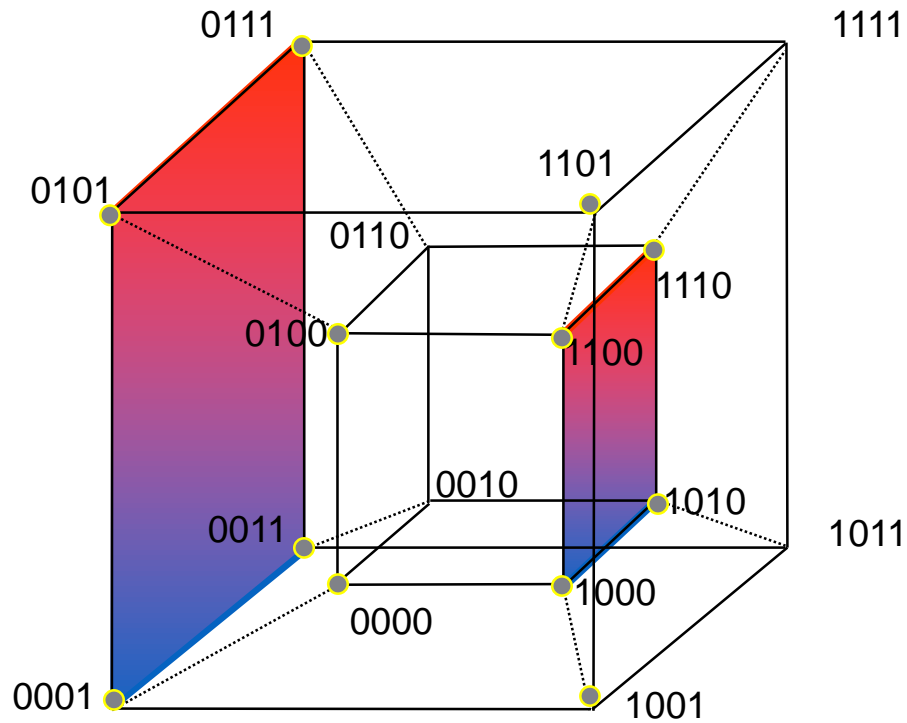
0 - 1 1

1 - 0 1

1 - 1 0

Alle Minterme von f sind Eckpunkte von Kanten, die Implikanten sind $\Rightarrow \mathcal{P}_{\text{rim}}(f) = \emptyset$.

BEISPIEL QUINE-MCCLUSKEY (7)



$$L_1\{x_1, x_2, x_4\};$$

→	0 0 - 1
→	1 0 - 0
→	0 1 - 1
→	1 1 - 0

$$L_1\{x_1, x_2, x_3\};$$

<u>0 0 0 -</u>
<u>0 1 0 -</u>
<u>1 0 0 -</u>
<u>1 1 0 -</u>

$$L_1\{x_2, x_3, x_4\};$$

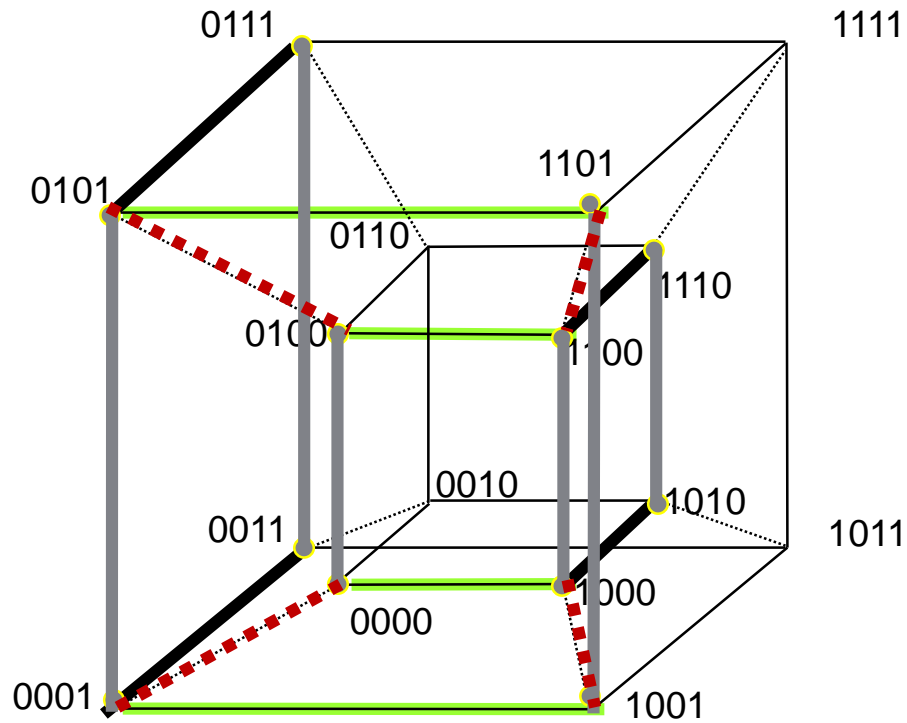
<u>- 0 0 0</u>
<u>- 0 0 1</u>
<u>- 1 0 0</u>
<u>- 1 0 1</u>

$$L_1\{x_1, x_3, x_4\};$$

<u>0 - 0 0</u>
<u>0 - 0 1</u>
<u>1 - 0 0</u>
<u>0 - 1 1</u>
<u>1 - 0 1</u>
<u>1 - 1 0</u>

Alle Implikanten aus $L_1\{x_1, x_2, x_4\}$ sind Kanten von Flächen, die Implikanten sind $\Rightarrow \text{Prim}(f) = \emptyset$.

BEISPIEL QUINE-MCCLUSKEY (8)



$L_1\{x1, x2, x4\}:$

0 0 - 1

1 0 - 0

0 1 - 1

1 1 - 0

$L_1\{x1, x2, x3\}:$

0 0 0 -

0 1 0 -

1 0 0 -

1 1 0 -

$L_1\{x2, x3, x4\}:$

- 0 0 0

- 0 0 1

- 1 0 0

- 1 0 1

$L_1\{x1, x3, x4\}:$

0 - 0 0

0 - 0 1

1 - 0 0

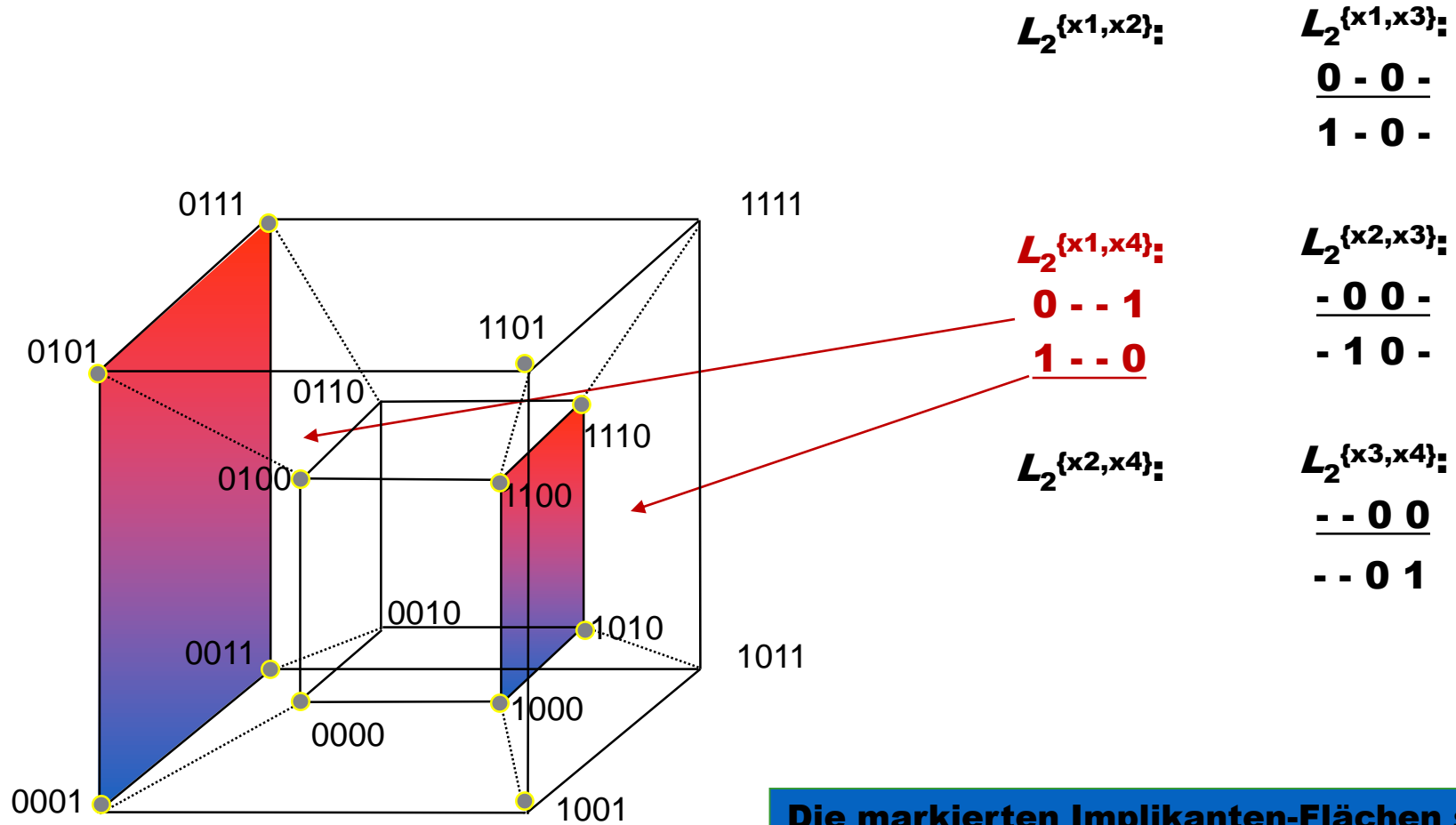
0 - 1 1

1 - 0 1

1 - 1 0

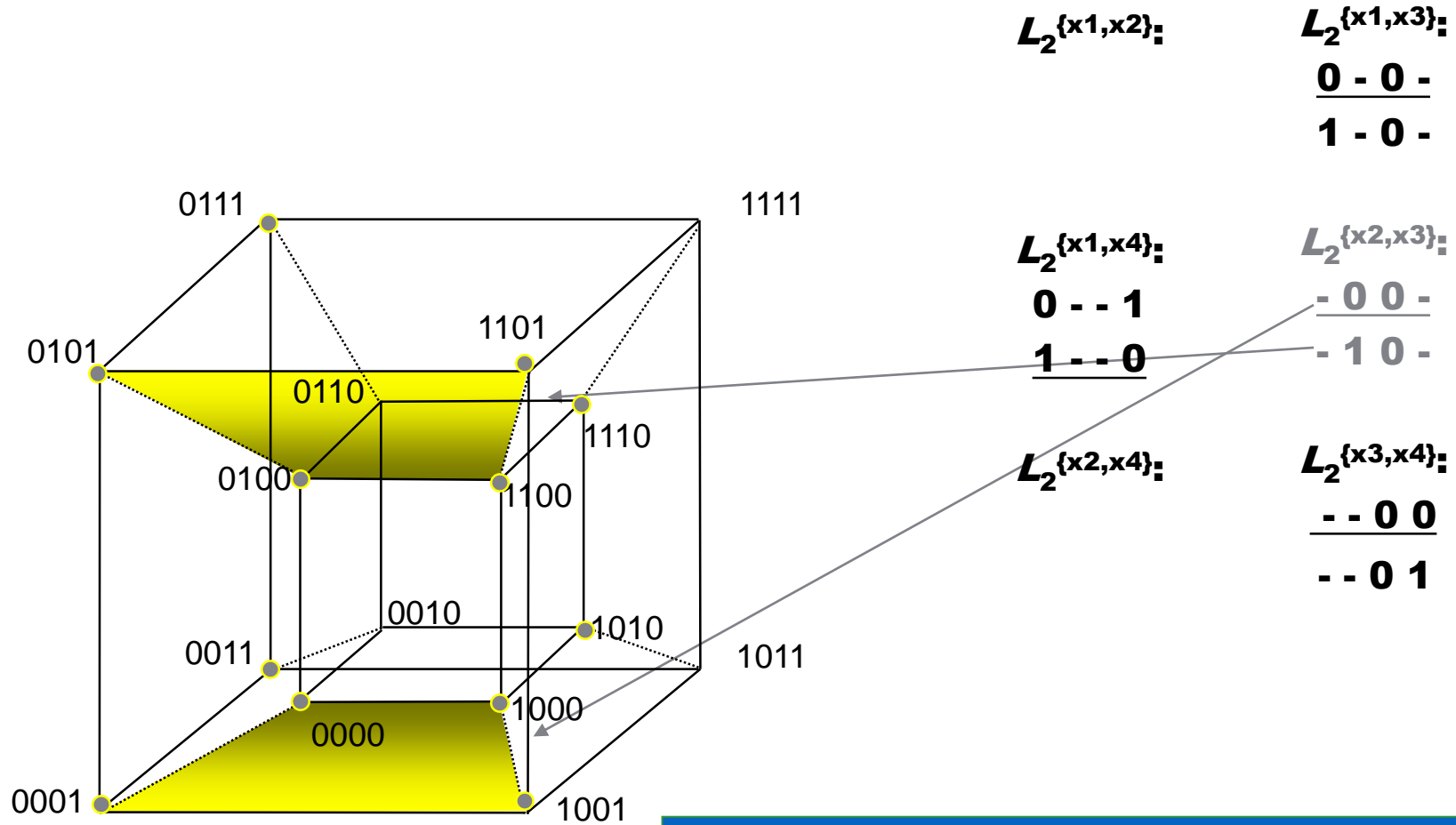
Alle Implikanten aus L_1^M sind Kanten von Flächen, die Implikanten sind $\Rightarrow \text{Prim}(f)=\emptyset$.

BEISPIEL QUINE-MCCLUSKEY (9)



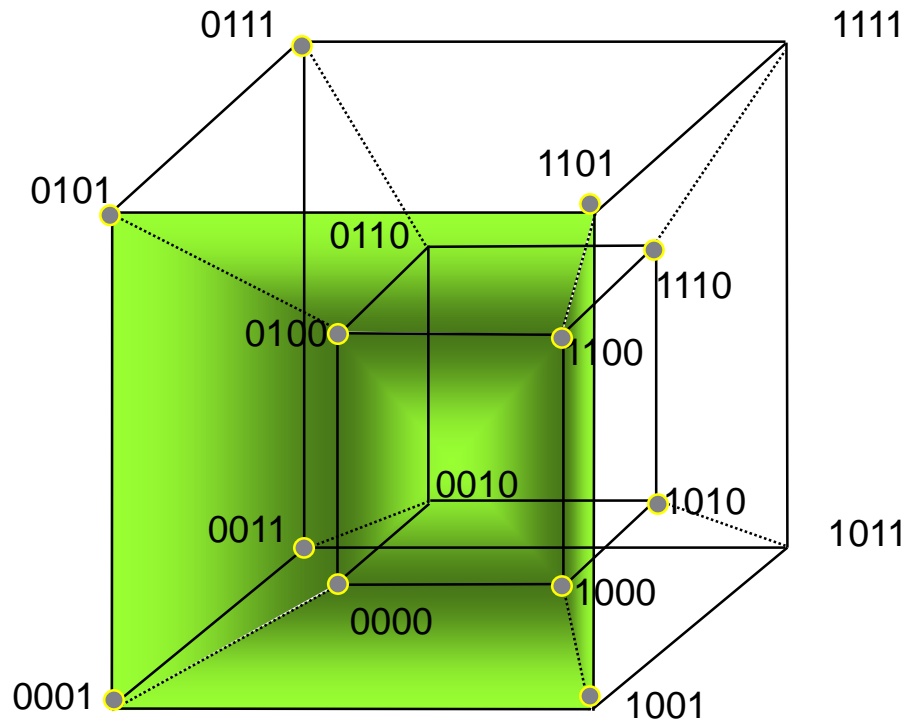
Die markierten Implikanten-Flächen sind nicht Rand eines 3-dim. Implikanten. Sie sind also prim! $\Rightarrow \text{Prim}(f) = \{x_1' x_4, x_1 x_4'\}$

BEISPIEL QUINE-MCCLUSKEY (10)



Die markierten Implikanten-Flächen sind Rand eines 3-dimensionalen Implikanten. Sie sind also nicht prim! $\Rightarrow \text{Prim}(f) = \{x_1'x_4, x_1x_4'\}$

BEISPIEL QUINE-MCCLUSKEY (11)



$L_3\{x1\}:$

$L_3\{x2\}:$

$L_3\{x3\}:$

$L_3\{x4\}:$

-- 0 -

$$\text{Prim}(f) = \{x_1' x_4, x_1 x_4', x_3'\}$$

$$\Rightarrow \text{Prim}(f) = \{x_1' x_4, x_1 x_4', x_3'\}$$

$$\Rightarrow p_{\text{complete}}(f) = x_1' x_4 + x_1 x_4' + x_3'$$

ZUSAMMENFASSUNG DES BEISPIELS

$L_0\{x_1, x_2, x_3, x_4\}:$

0 0 0 0

0 0 0 1

0 1 0 0

1 0 0 0

0 0 1 1

0 1 0 1

1 0 0 1

1 0 1 0

1 1 0 0

0 1 1 1

1 1 0 1

1 1 1 0

$L_1\{x_1, x_2, x_4\}:$

0 0 - 1

1 0 - 0

0 1 - 1

1 1 - 0

$L_1\{x_2, x_3, x_4\}:$

- 0 0 0

- 0 0 1

- 1 0 0

- 1 0 1

$L_1\{x_1, x_2, x_3\}:$

0 0 0 -

0 1 0 -

1 0 0 -

1 1 0 -

$L_1\{x_1, x_3, x_4\}:$

0 - 0 0

0 - 0 1

1 - 0 0

0 - 1 1

1 - 0 1

1 - 1 0

$L_2\{x_1, x_2\}:$

$L_2\{x_1, x_4\}:$

0 - - 1

1 - - 0

$L_2\{x_2, x_4\}:$

$L_2\{x_1, x_3\}:$

0 - 0 -

1 - 0 -

$L_2\{x_2, x_3\}:$

- 0 0 -

- 1 0 -

$L_2\{x_3, x_4\}:$

- - 0 0

- - 0 1

$L_3\{x_1\}:$

$L_3\{x_3\}:$

Nicht kürzbar
→ prim!

- - 0 -

$$\Rightarrow p_{complete}(f) = x_1' x_4 + x_1 x_4' + x_3'$$

LÖSEN DES ÜBERDECKUNGSPROBLEMS (1)

Sei im Folgenden die Menge $\text{Prim}(f)$ der Primimplikanten von f gegeben.

Gesucht

ist eine kostenminimale Teilmenge M von $\text{Prim}(f)$,
mit deren Monomen die Funktion f beschreibbar ist.

LÖSEN DES ÜBERDECKUNGSPROBLEMS (2)

Definiere eine Boolesche Matrix $PIT(f)$, die **Primimplikantentafel** von f

- die **Zeilen** entsprechen eindeutig den **Primimplikanten** von f
- die **Spalten** entsprechen eindeutig den **Mintermen** von f
- Sei $min(\alpha)$ ein beliebiger Minterm von f . Dann gilt für PI m :
$$PIT(f) [m, min(\alpha)] = 1 \Leftrightarrow \psi(m)(\alpha) = 1$$

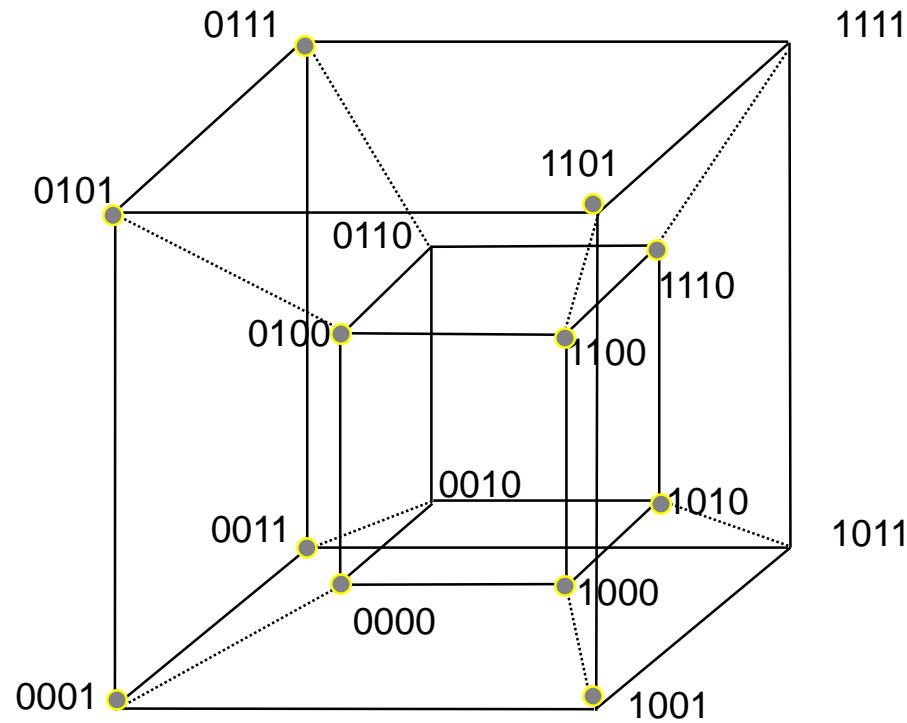
Der Eintrag an der Stelle $[m, min(\alpha)]$ ist also genau dann 1, wenn $min(\alpha)$ eine Ecke des Würfels m beschreibt.

Gesucht

eine kostenminimale Teilmenge M von $\mathcal{P}rim(f)$,
so dass jede Spalte von $PIT(f)$ überdeckt ist,

d.h. $\forall \alpha \in ON(f) \exists m \in M$ mit $PIT(f)[m, min(\alpha)] = 1$.

LÖSEN DES ÜBERDECKUNGSPROBLEMS (3)



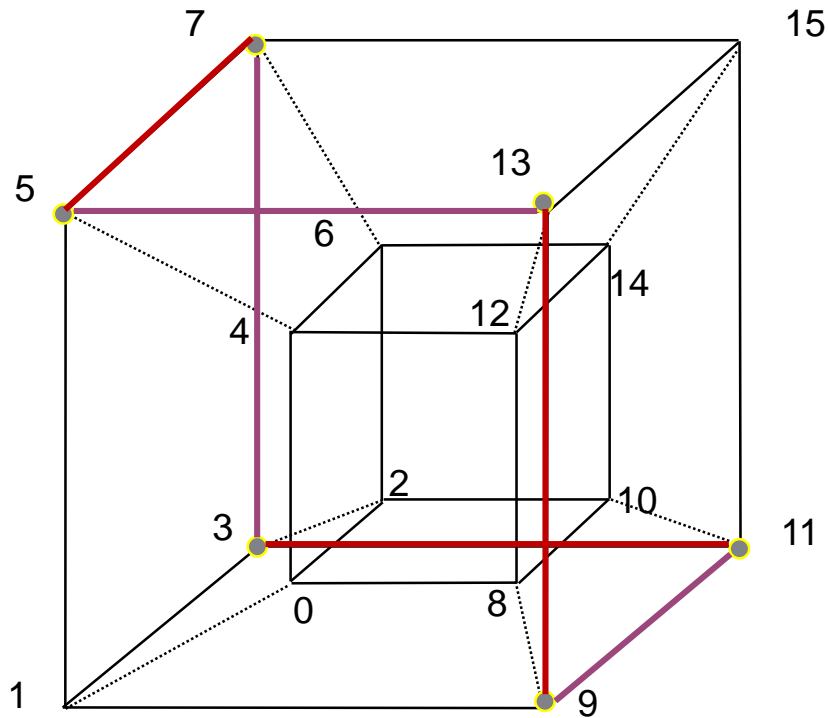
Primimplikantentafel ***PIT(f)***:

	0	1	3	4	5	7	8	9	10	12	13	14
$x_1' x_4$	1	1			1	1						
$x_1 x_4'$							1		1	1		1
x_3'	1	1		1	1		1	1		1	1	

⇒ **Alle Primimplikanten sind wesentlich!**

DAS MATRIX-ÜBERDECKUNGSPROBLEM (4)

Ein anderes Beispiel



Primimplikantentafel $PIT(f)$:

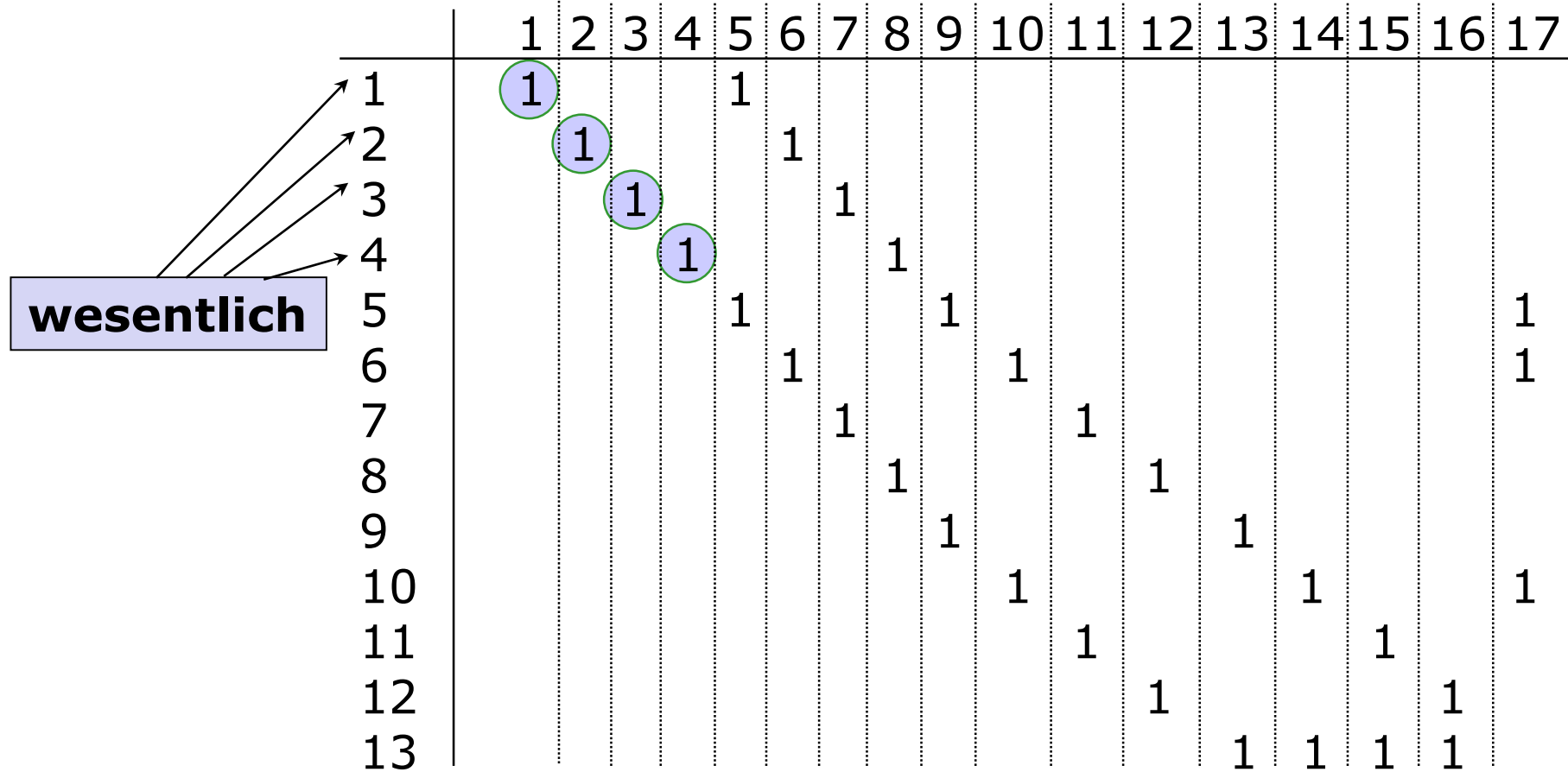
	3	5	7	9	11	13
$\{7,5\}$		1	1			
$\{5,13\}$		1				1
$\{13,9\}$				1		1
$\{9,11\}$				1	1	
$\{11,3\}$	1				1	
$\{3,7\}$	1		1			

Kein Primimplikant ist wesentlich!

REDUKTIONSREGELN

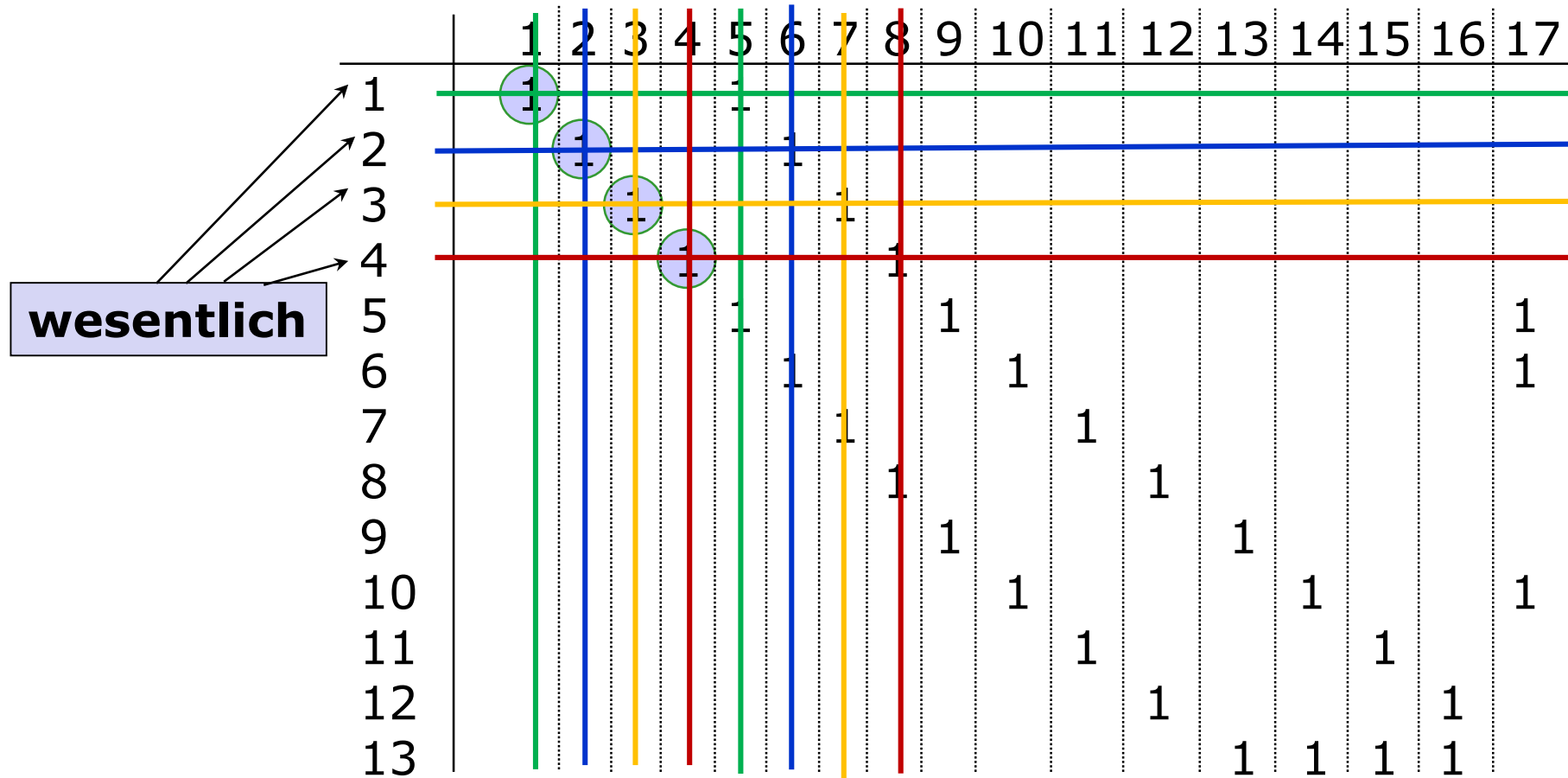
1. Entferne aus der Primimplikantentafel $PIT(f)$ alle wesentlichen Primimplikanten und alle Minterme, die von diesen überdeckt werden.
2. Entferne aus der Primimplikantentafel $PIT(f)$ alle Minterme, die einen anderen Minterm in $PIT(f)$ dominieren.
3. Entferne aus $PIT(f)$ alle Primimplikanten, die durch einen anderen, nicht teureren Primimplikanten dominiert werden.

ERSTE REDUKTIONSREGEL



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1				1												
2		1				1											
3			1				1										
4				1				1									
5					1				1								1
6						1				1							1
7							1				1						
8								1				1					
9									1				1				
10										1				1			1
11											1				1		
12												1				1	
13													1	1	1	1	

ERSTE REDUKTIONSREGEL



ERSTE REDUKTIONSREGEL

Überdeckungsproblem nach Anwendung der ersten Reduktionsregel:

	9	10	11	12	13	14	15	16	17
5	1								1
6		1							1
7			1						
8				1					
9	1				1				
10		1				1			1
11			1				1		
12				1				1	
13					1	1	1	1	

**Die Matrix enthält
keine wesentlichen
Zeilen mehr!**

ZWEITE REDUKTIONSREGEL

Definition

Es sei A eine Boolesche Matrix. Spalte j der Matrix A dominiert Spalte i der Matrix A , wenn $A[k,i] \leq A[k,j]$ für jede Zeile k gilt.

Nutzen für unser Problem

Dominiert ein Minterm w' von f einen anderen Minterm w von f , so braucht man w' nicht mehr weiter zu betrachten, da w auf jeden Fall überdeckt werden muss und hierdurch auch Minterm w' überdeckt wird.

Jeder noch in $PIT(f)$ vorhandene Primimplikant p , der w überdeckt, überdeckt auch w' .

ZWEITE REDUKTIONSREGEL

	9	10	11	12	13	14	15	16	17
5	1								1
6		1							1
7			1						
8				1					
9	1				1				
10		1				1			1
11			1				1		
12				1				1	
13					1	1	1	1	

Spalte 17 dominiert Spalte 10
 ⇒ **Spalte 17 kann gelöscht werden!**

DRITTE REDUKTIONSREGEL

Definition

Sei A eine Boolesche Matrix. Zeile i der Matrix A **dominiert** Zeile j der Matrix A , wenn $A[i,k] \geq A[j,k]$ für jede Spalte k gilt.

Nutzen für unser Problem

Dominiert ein Primimplikant m einen Primimplikanten m' , so braucht man m' nicht mehr weiter zu betrachten, wenn $\text{cost}(m') \geq \text{cost}(m)$ gilt.

Der Primimplikant m überdeckt jeden noch nicht überdeckten Minterm von f , der von m' überdeckt wird, obwohl er nicht teurer ist.

Dritte Reduktionsregel (2)

Nehme an, dass die Zeilen 5 bis 12 gleiche Kosten haben.

	9	10	11	12	13	14	15	16
5	1							
6		1						
7			1					
8				1				
9	1				1			
10		1				1		
11			1				1	
12				1				1
13					1	1	1	1

werden dominiert

DRITTE REDUKTIONSREGEL (2)

Nehme an, dass die Zeilen 5 bis 12 gleiche Kosten haben.

	9	10	11	12	13	14	15	16
5	1							
6		1						
7			1					
8				1				
9	1				1			
10		1				1		
11			1				1	
12				1				1
13					1	1	1	1

DRITTE REDUKTIONSREGEL (3)

Überdeckungsproblem
nach Anwendung der
dritten Reduktionsregel:

	9	10	11	12	13	14	15	16
9	1				1			
10		1				1		
11			1				1	
12				1				1
13					1	1	1	1

Offensichtlich kann nun wieder die erste Reduktionsregel angewendet werden, da die Zeilen 9, 10, 11, 12 wesentlich sind.

⇒ Die resultierende Matrix ist leer

⇒ Das gefundene Minimalpolynom ist $1+2+3+4+9+10+11+12$

... enthält nicht (wie erwartet) die Zeile mit der maximalen Anzahl Einsen!

ZYKLISCHE ÜBERDECKUNGSPROBLEME

Definition

Eine Primimplikantentafel heißt **reduziert**, wenn keine der drei Reduktionsregeln anwendbar ist.

Ist eine reduzierte Tafel nicht-leer, spricht man von einem **zyklischen Überdeckungsproblem**.

Primimplikantentafel **PIT(f)**:

	3	5	7	9	11	13
{7,5}		1	1			
{5,13}		1				1
{13,9}				1		1
{9,11}				1	1	
{11,3}	1				1	
{3,7}	1		1			

LÖSUNG DES ZYKLISCHEN ÜBERDECKUNGSPROBLEMS

Verfahren von Petrick

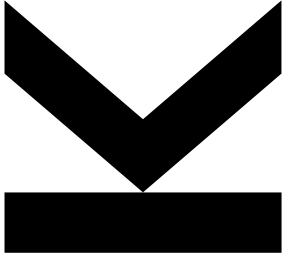
- Übersetze die PIT in eine Produktsumme, d.h. in ein (OR,AND)-Polynom, das alle Möglichkeiten der Überdeckung enthält.
- Multipliziere die Produktsumme aus, so dass ein (AND-OR) Polynom entsteht.
- Die gesuchte minimale Überdeckung ist gegeben durch das **kürzeste Monom**

	1	2	3	4
1	1	1		
2			1	1
3	1		1	
4		1		1
5	1			1
6		1	1	

wird übersetzt in

$$\begin{aligned}
 & (1+3+5)(1+4+6)(2+3+6)(2+4+5) \\
 &= (1+14+16+13+34+36+15+45+56) * \\
 & \quad (2+24+25+23+34+35+26+46+56) \\
 &= \mathbf{12+124+125+123+134+\dots+34+\dots+56}
 \end{aligned}$$

BINÄRE ENTSCHEIDUNGSDIAGRAMME



Robert Wille (robert.wille@jku.at)

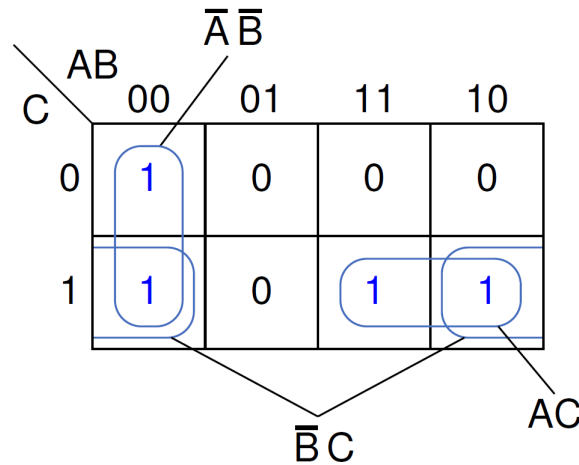
Sebastian Pointner (sebastian.pointner@jku.at)

Institut für Integrierte Schaltungen

Abteilung für Schaltkreis- und Systementwurf

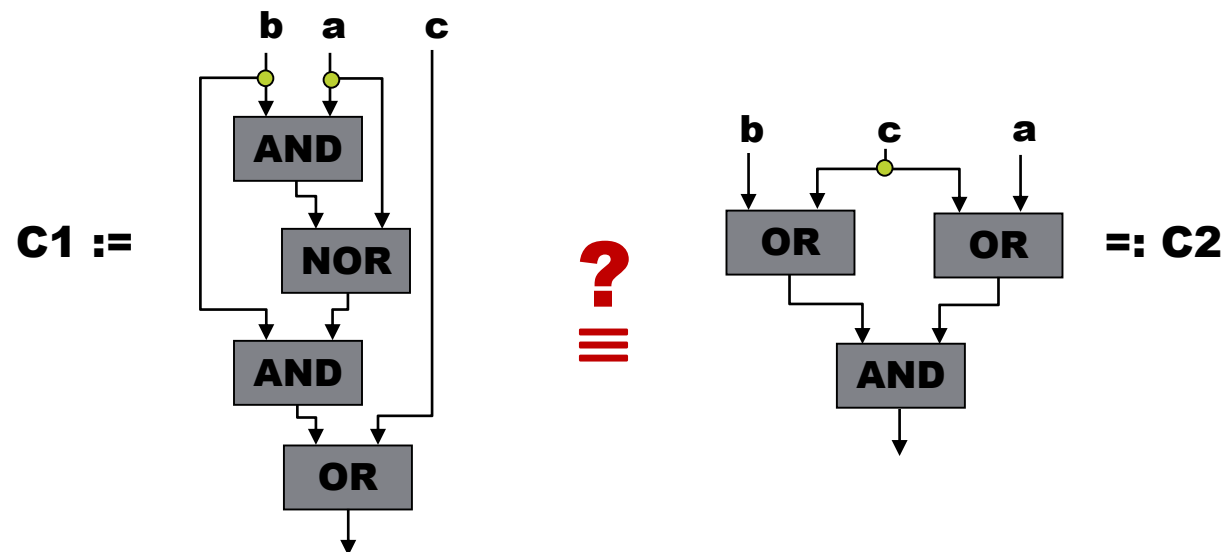
PROBLEME BISHERIGER VERFAHREN

■ Komplexität/Skalierbarkeit



x_1	x_2	x_3	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	0
1	1	1	1

■ Vergleichbarkeit



PROBLEME BISHERIGER VERFAHREN

■ Komplexität/Skalierbarkeit

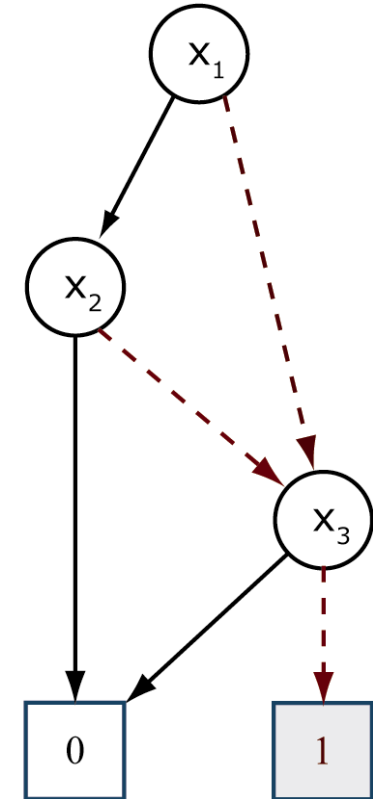
■ Ziel:

- ☐ Gelungener Kompromiss zwischen kompakter Darstellung und effizienter Manipulation
- ☐ Kanonische Darstellung

■ Vergleichbarkeit

BINÄRES ENTSCHEIDUNGSDIAGRAMM

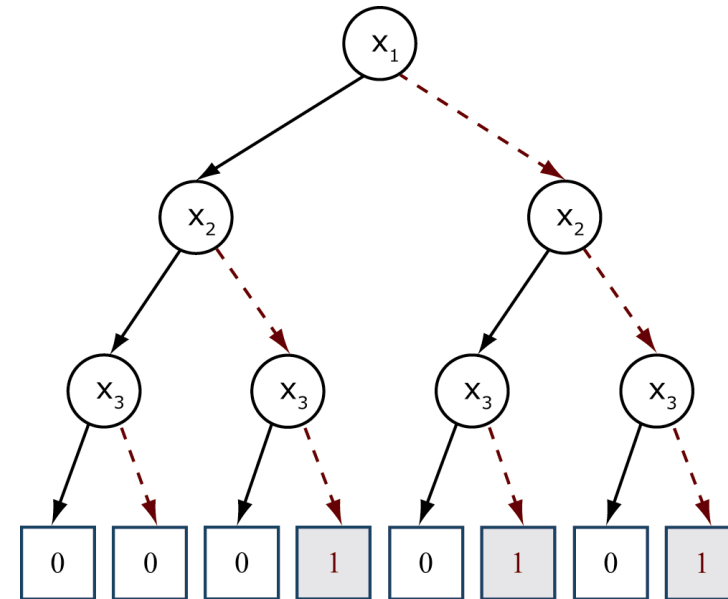
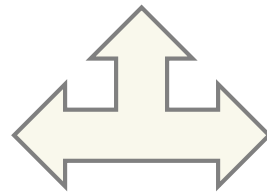
- azyklischer Graph mit einer Wurzel
- Innere Knoten
 - sind mit einer Variablen markiert
 - haben zwei Nachfolger,
 - das low-Kind (hier: gestrichelte Kante) und
 - das high-Kind (hier: durchgezogene Kante)
- Blätter/Terminale Knoten
 - sind mit einer Konstanten (0 oder 1) markiert
- Jeder Knoten repräsentiert eine (Teil-)Funktion
 - Blatt: Konstante Funktion, die jede Eingabe auf 0/1 abbildet
 - Innere Knoten (markiert mit x_j): $f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$



ILLUSTRATION

x_1	x_2	x_3	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	0
1	1	1	1

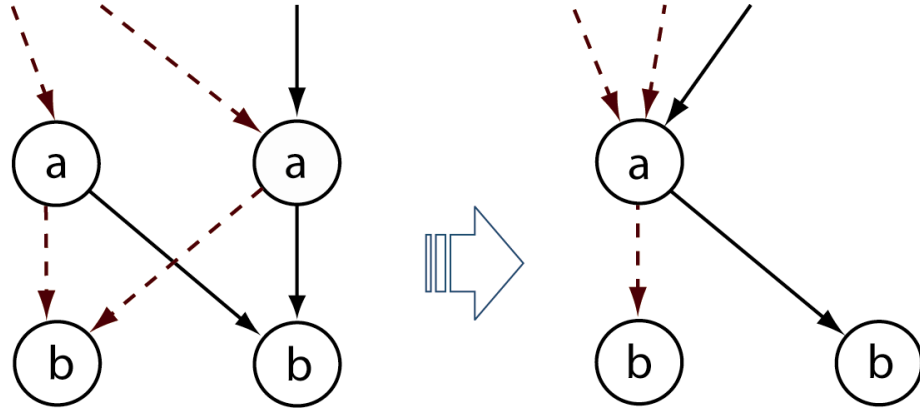
x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
F	0	0	0	1	0	1	0	1



REDUKTIONSREGELN

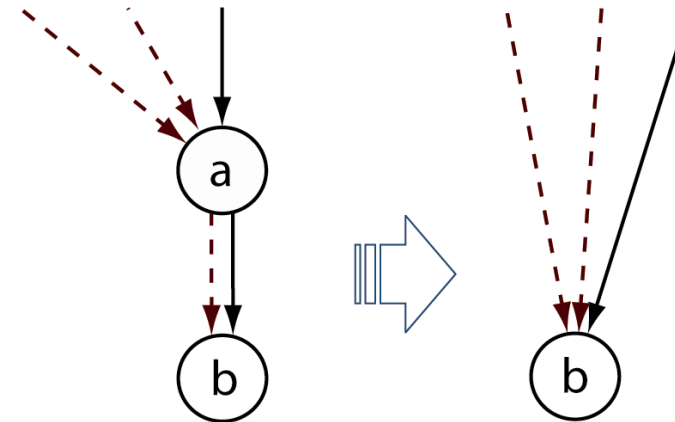
Regel 1: Isomorphismus

Führe isomorphe Knoten zusammen.
(I-Reduction)

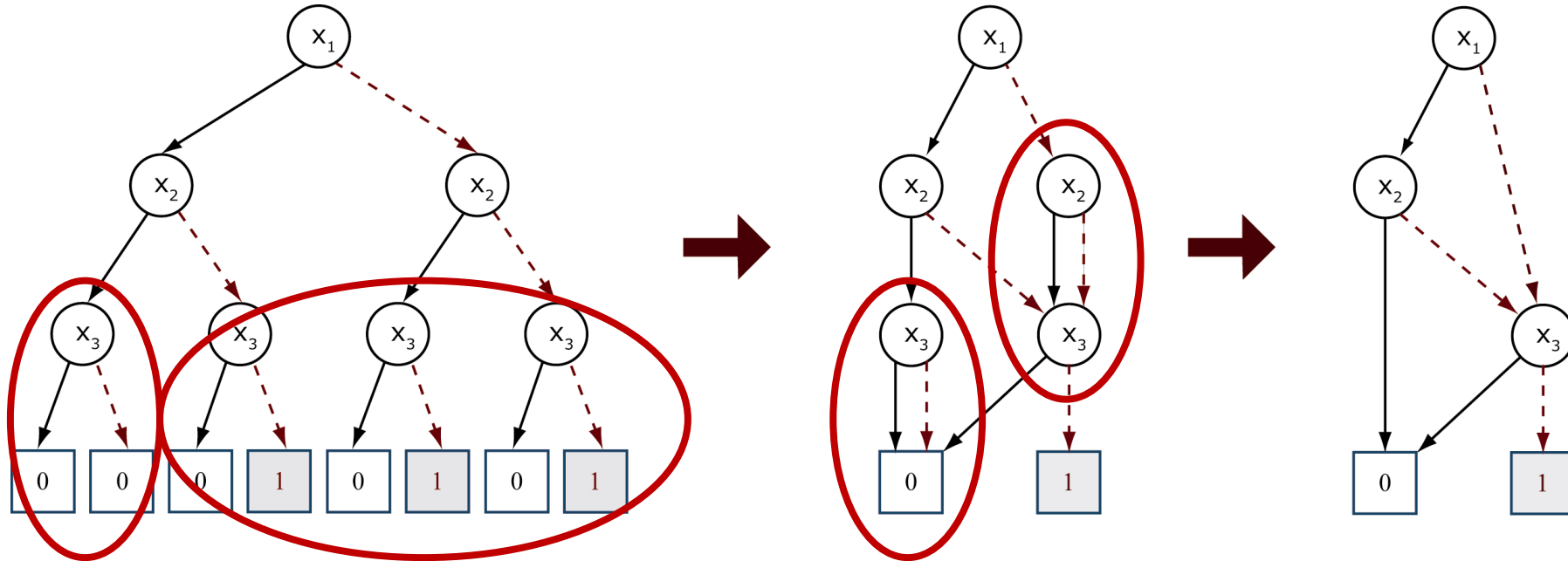


Regel 2: Elimination

Entferne Knoten, dessen beide Kinder zum gleichen Nachfolger zeigen.
(S-Reduktion)



ANWENDUNG DER REDUKTIONSREGELN



WAS PASSIERT KONKRET?

■ Funktion wird in Teilfunktionen zerlegt

■ Kofaktor von f

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

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

■ Shannon Zerlegung

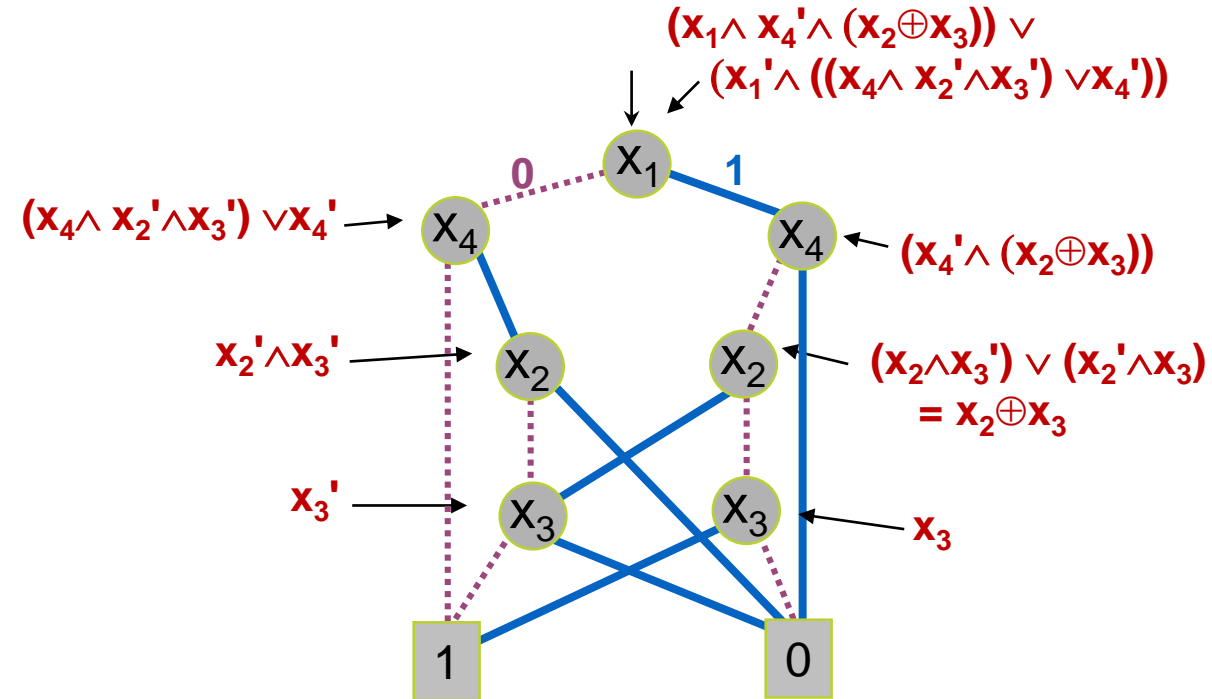
$$f = (x_i \wedge f_{x_i=1}) \vee (x_i' \wedge f_{x_i=0})$$

■ Weitere Zerlegungstypen (im Folgenden nicht weiter betrachtet):

□ Positiv Davio: $f = f_{x_i=0} \oplus x_i \wedge (f_{x_i=0} \oplus f_{x_i=1})$

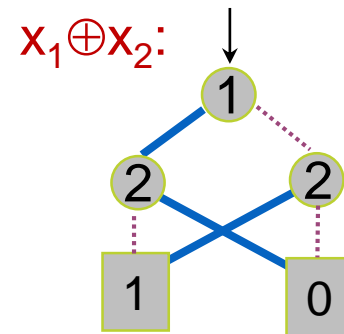
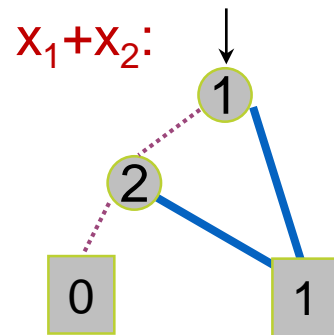
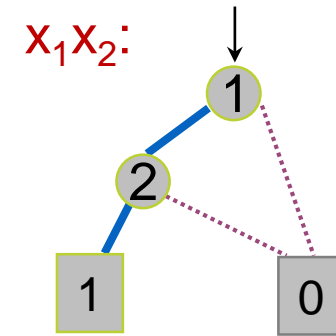
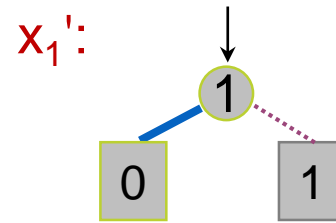
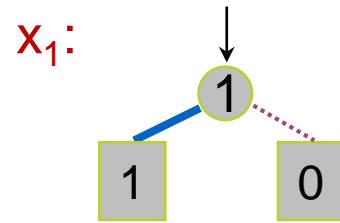
□ Negativ Davio: $f = f_{x_i=1} \oplus x_i' \wedge (f_{x_i=0} \oplus f_{x_i=1})$

BEISPIEL



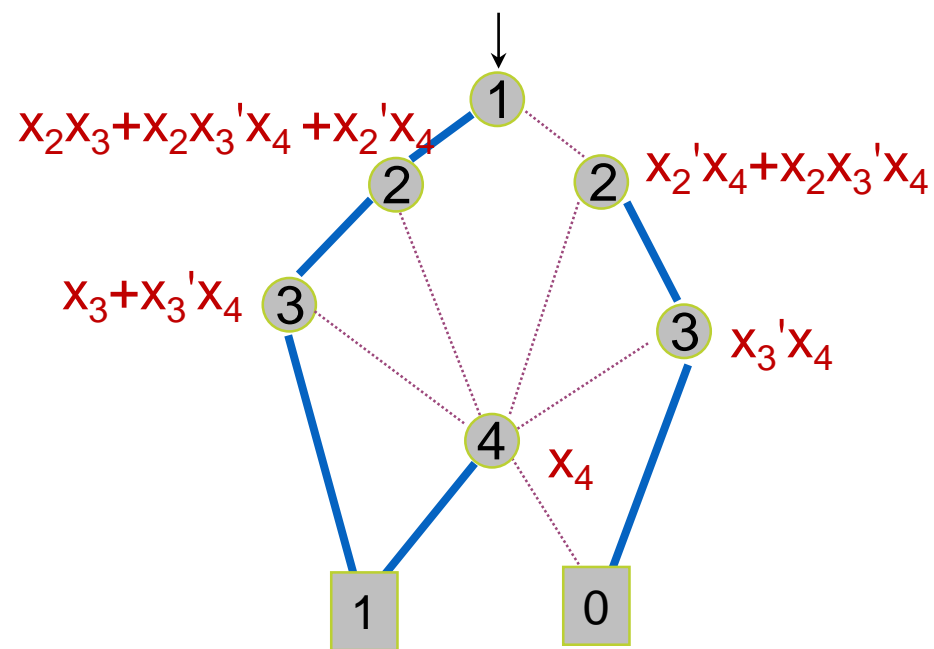
$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

BDDs VON TYPISCHEN FUNKTIONEN

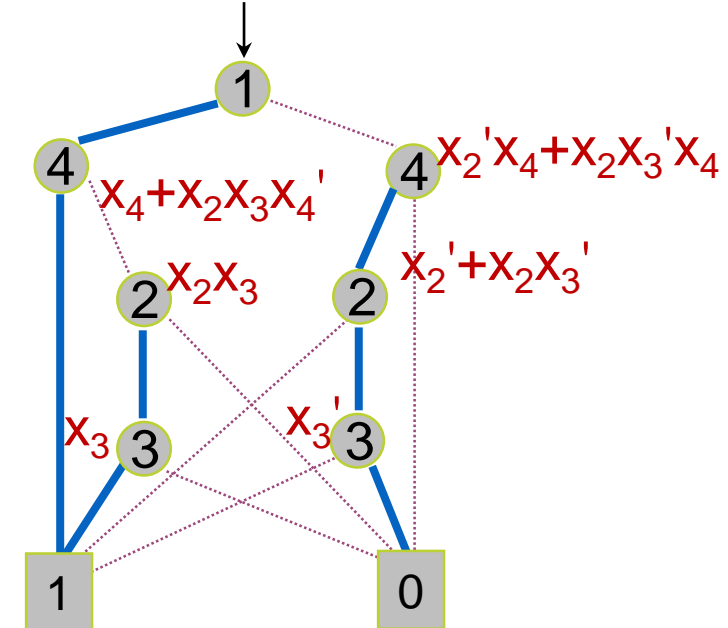


WEITERE BEISPIELE FÜR BDDS

$$x_1x_2x_3 + x_1x_2x_3'x_4 + x_1x_2'x_4 + x_1'x_2'x_4 + x_1'x_2x_3'x_4$$



$$x_1x_4 + x_1x_2x_3x_4' + x_1'x_2'x_4 + x_1'x_2x_3'x_4$$

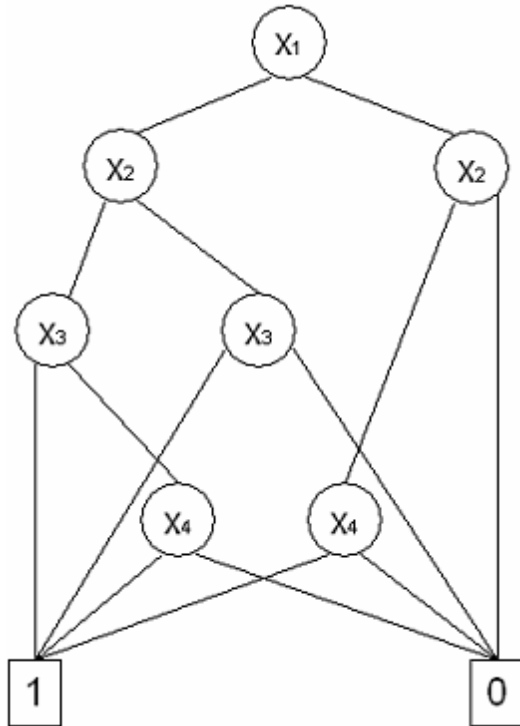


... beschreiben beide die Boolesche Funktion $x_1x_2x_3 + x_2'x_4 + x_3'x_4$

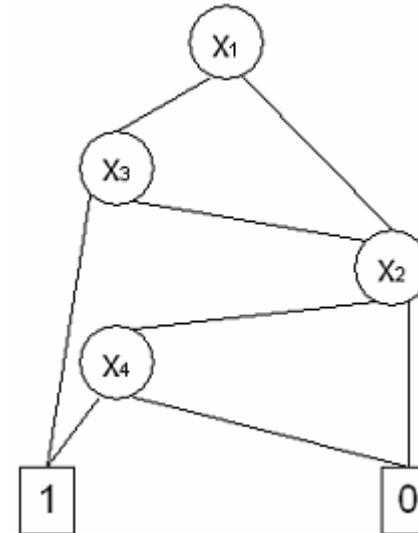
GEORDNETE ENTSCHEIDUNGSDIAGRAMME

- Ein Entscheidungsdiagramm heißt **frei**, wenn auf jedem Pfad von der Wurzel zu einem Blatt jede Variable höchstens einmal als Markierung vorkommt.
 - Ein Entscheidungsdiagramm heißt **geordnet**, wenn auf jedem Pfad von der Wurzel zu einem Blatt die Variablen in der gleichen Reihenfolge abgefragt werden.
 - Ein Entscheidungsdiagramm heißt **reduziert**, wenn sich keine Reduktionsregeln mehr anwenden lassen.
- Reduced Ordered Binary Decision Diagrams (ROBDDs)

EFFEKT DER VARIABLENORDNUNG



$$f = x_1x_3 + x_2x_4$$



FINDEN DER BESTEN VARIABLENORDNUNG

- Komplexitätstheoretisch schwierig
 - ☐ NP-vollständig
 - ☐ Es gibt (nur laufzeitintensive) exakte Algorithmen!

- Heuristiken zur BDD-Minimierung
 - ☐ Initiale Verfahren
 - bauen ausgehend von einer anderen Darstellung einen ersten BDD auf
 - ☐ Umordnungsverfahren
 - versuchen, eine bestehende Variablenordnung zu verbessern
 - ☐ Beispiel: Sifting

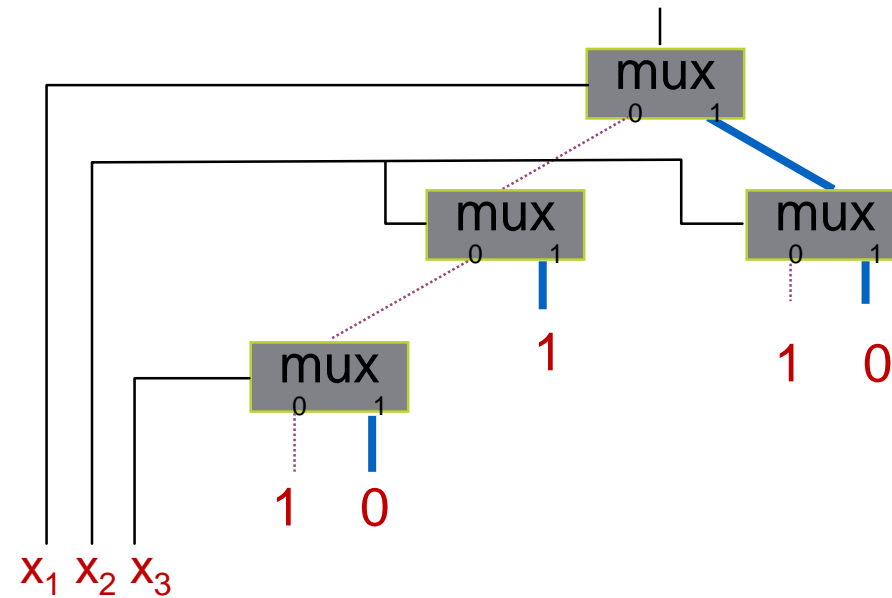
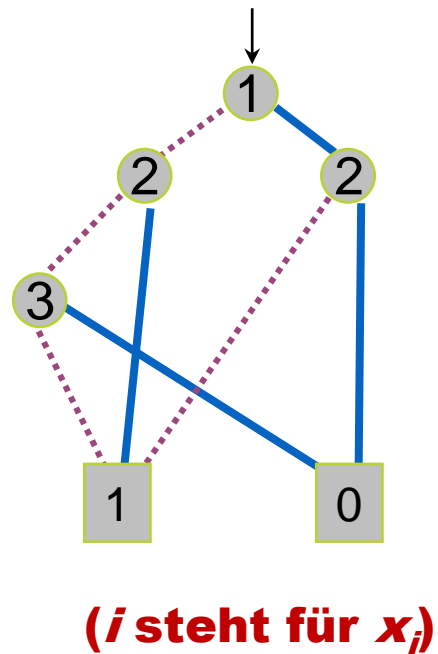
GRÖÖE VON BDDS

- In schlechtesten Fall: exponentiell
(gilt für die meisten Booleschen Funktionen)
- Aber deutlich besser für viele praktisch relevante Funktionen, z.B.
 - ☐ Symmetrische Functions (polynomiell)
 - ☐ Addierer (linear)
 - ☐ AND, OR, ... (linear)

KANONIZITÄT VON BDD: SATZ VON BRYANT (1986)

**Geordnete reduzierte binäre Entscheidungsdiagramme
sind kanonische Darstellungen Boolescher Funktionen.**

SYNTHESE



... beschreibt die Boolesche Funktion $x_1'x_2'x_3' + x_1'x_2 + x_1x_2'$

VERIFIKATION

