Kapitel 8: Zweistufige Logiksynthese

PLAs und zweistufige Logiksynthese

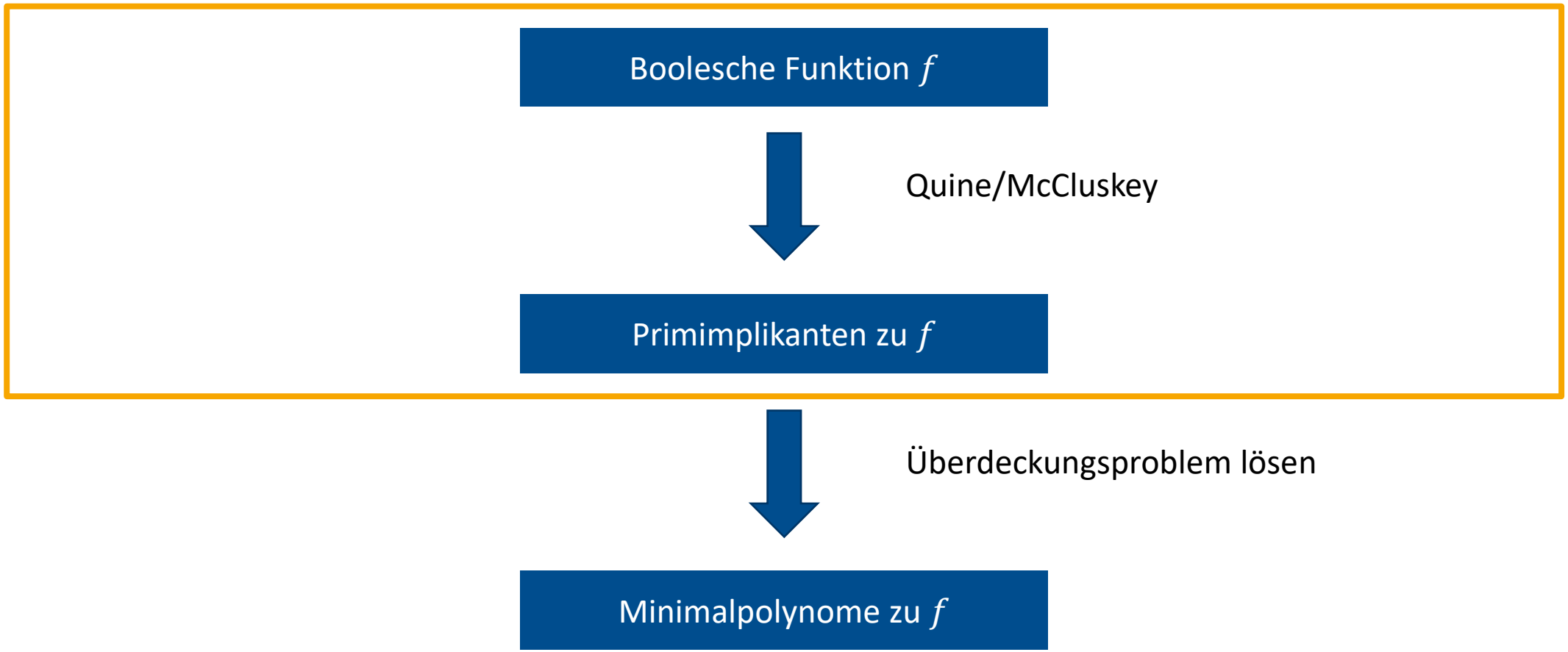
Implikanten und Primimplikanten

Algorithmus zur Berechnung eines Minimalpolynoms

Lernziele

- Algorithmus von Quine/McCluskey zur Bestimmung eines Minimalpolynoms kennen, verstehen und anwenden können
- Korrektheit von Quine/McCluskey nachvollziehen können
- Kosten von Quine/McCluskey kennen und verstehen, Beschleunigungsmöglichkeiten kennen
- Matrixüberdeckungsproblem kennen und verstehen
- Drei Reduktionsregeln zur Bestimmung wesentlicher Primimplikanten kennen, verstehen und anwenden können
- Zyklische Überdeckungsprobleme erkennen und mit Verfahren von Petrick lösen können

Algorithmus zur Berechnung eines Minimalpolynoms



Identifikation von Primimplikanten: Das Verfahren von Quine

polynom function Quine ($f: \mathbb{B}_n \rightarrow \mathbb{B}$)

begin

$L_0 := Minterm(f);$

$i := 0;$

$Prim(f) := \emptyset;$

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

loop

$L_{i+1} := \{m | \exists j \in \{1, \dots, n\}: \{mx_j, m\overline{x_j}\} \subset L_i\};$

$Prim(f) := Prim(f) \cup \{m | m \in L_i \; und \; m \; wird \; von \; keinem \; q \in L_{i+1} \; überdeckt\}$

$i := i + 1;$

end loop

return $Prim(f) \cup L_i$

end

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

Identifikation von Primimplikanten: Die Verbesserung von 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

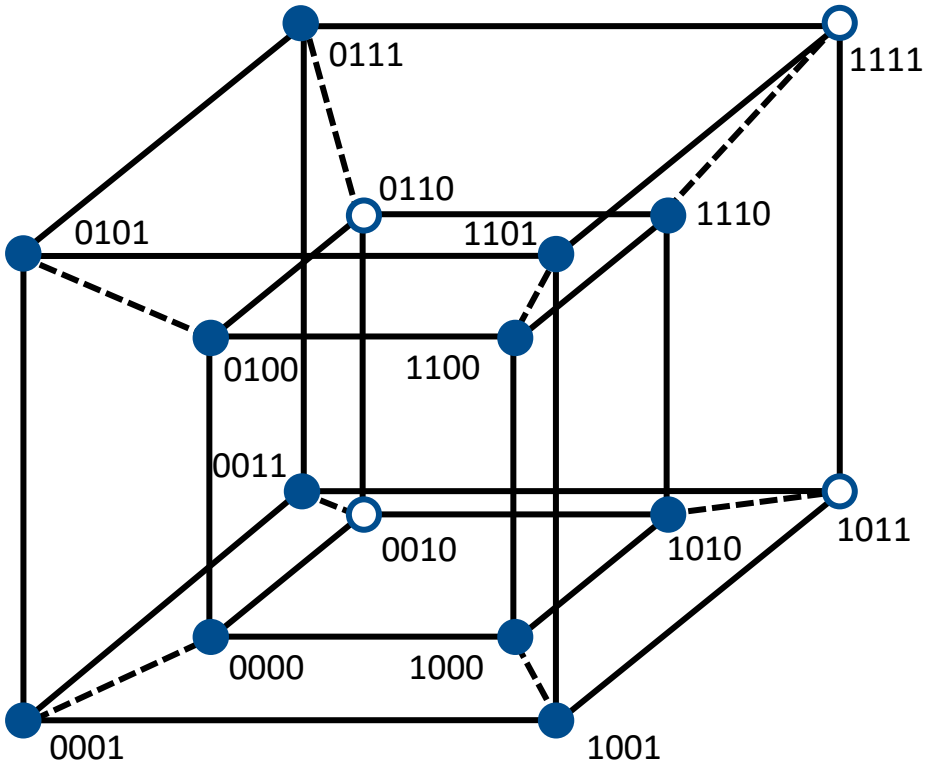
- Partitioniere L_i in Klassen L_i^M, mit $M \subseteq \{x_1, \dots x_n\}$ und $|M| = n - i$. L_i^M enthalte die Implikanten aus L_i, deren Literale alle aus M sind.
- Ordne die Monome in L_i^M gemäß der Anzahl der positiven Literale.

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

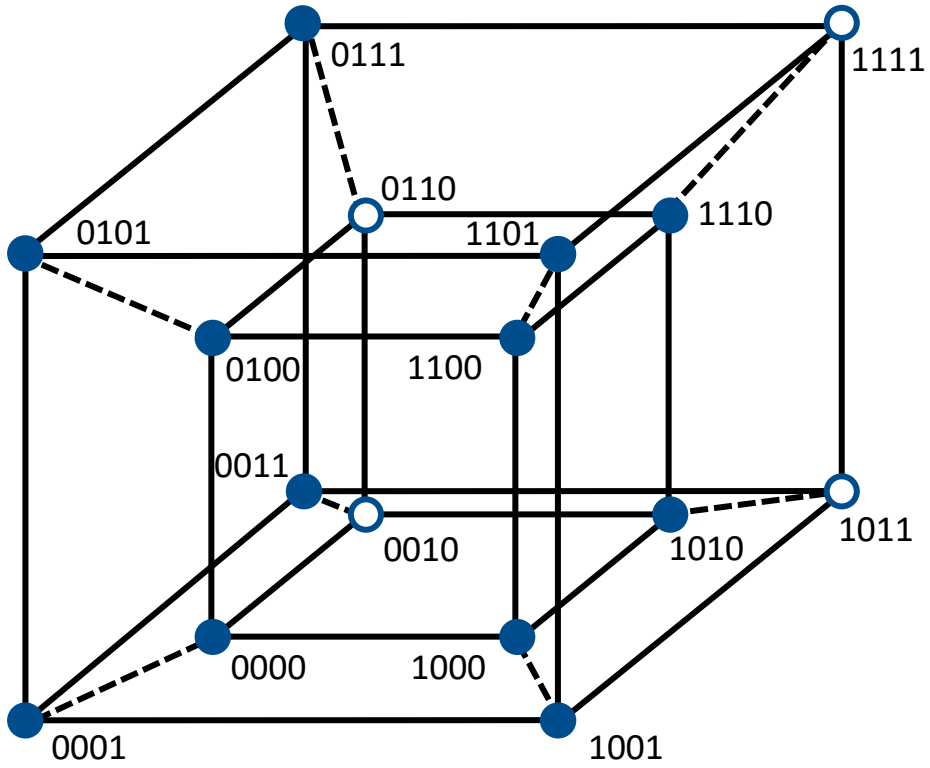
0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

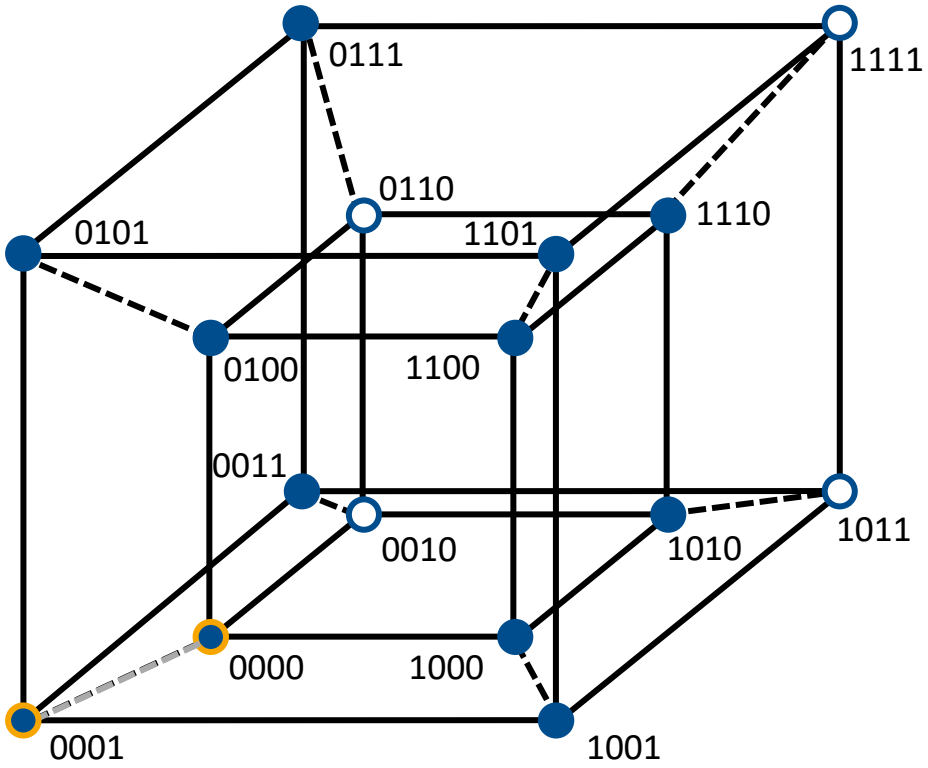
L_1: $\mathbf{0 \leftrightarrow 1}$

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001	0011	0111
	0100	0101	1101
	1000	1001	1110
		1010	
		1100	

L_1:

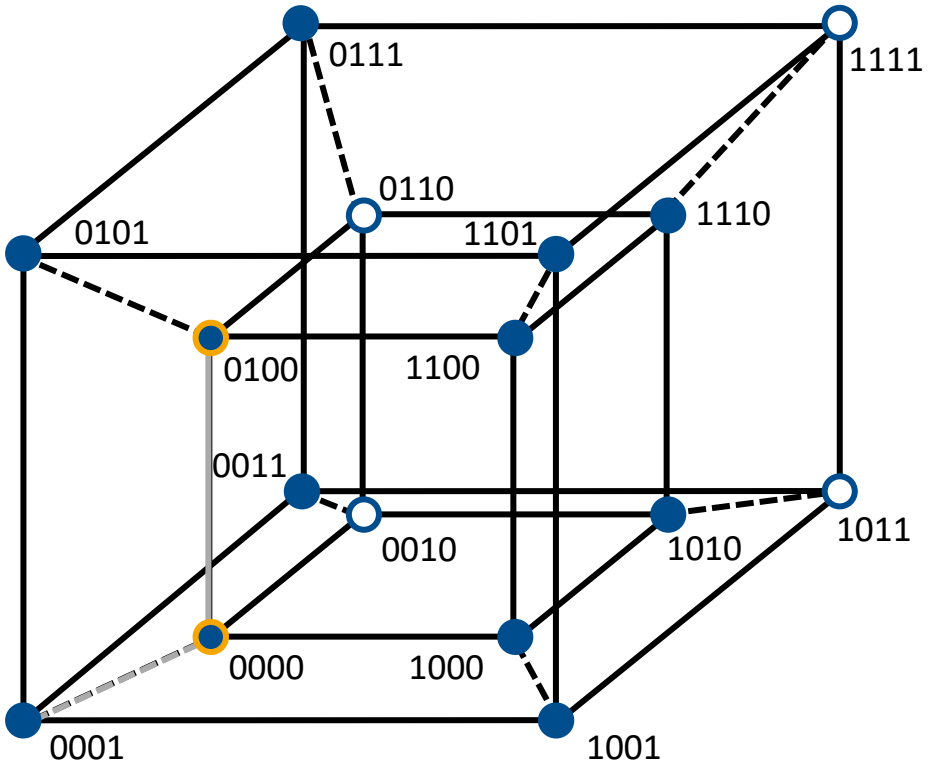
$0 \leftrightarrow 1$
000–

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001	0011	0111
	0100	0101	1101
	1000	1001	1110
		1010	
		1100	

L_1: $\mathbf{0 \leftrightarrow 1}$

000–

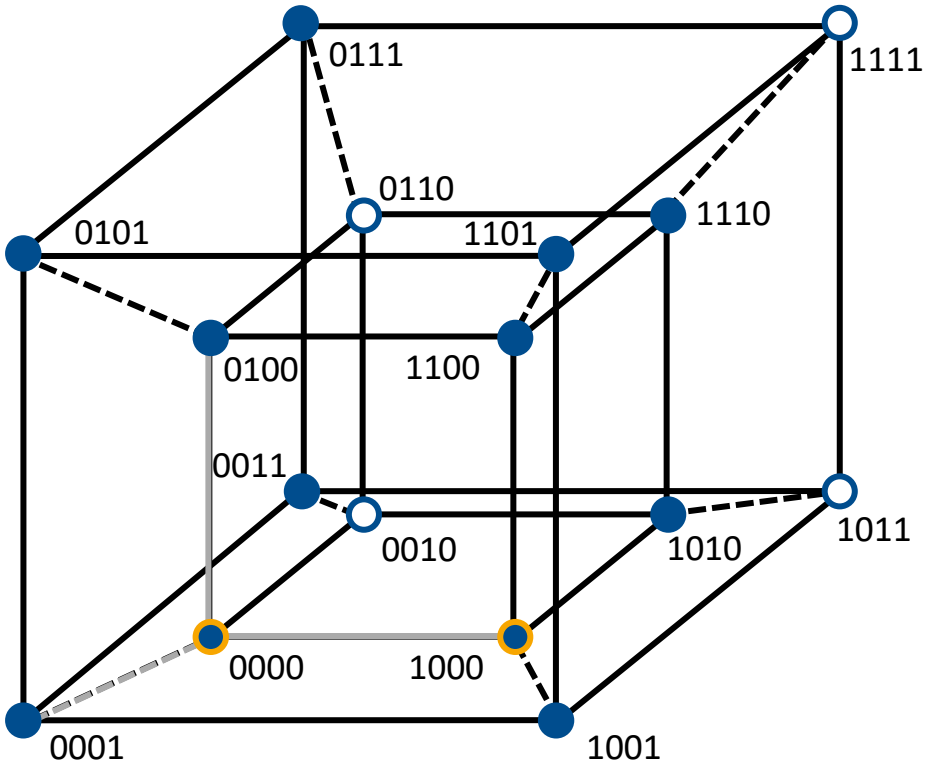
0–00

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001	0011	0111
	0100	0101	1101
	1000	1001	1110
		1010	
		1100	

L_1: $\mathbf{0 \leftrightarrow 1}$

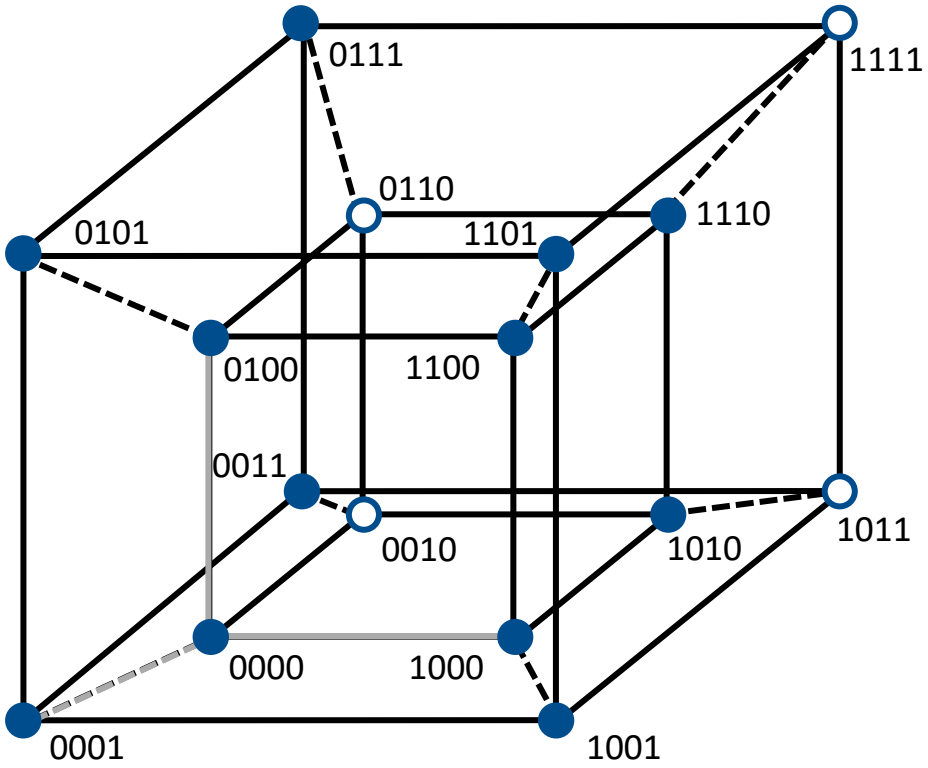
$000-$
$0-00$
-000

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

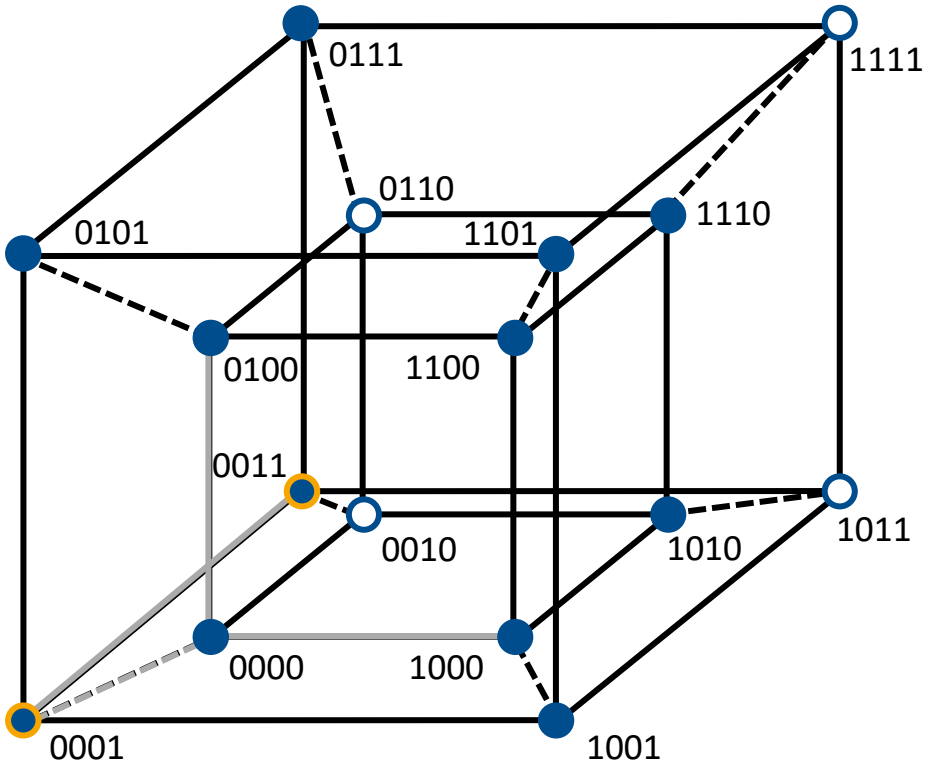
$0 \leftrightarrow 1$	$1 \leftrightarrow 2$
000− 0−00 −000	

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

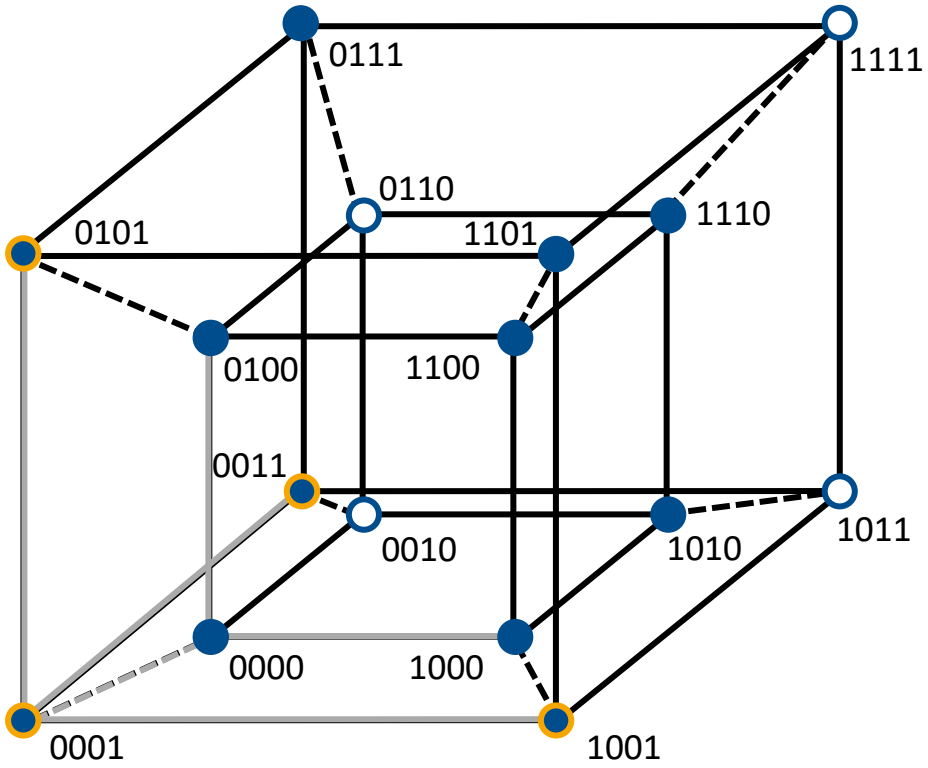
$0 \leftrightarrow 1$	$1 \leftrightarrow 2$
$000-$	$00-1$
$0-00$	
-000	

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001	0011	0111
	0100	0101	1101
	1000	1001	1110
		1010	
		1100	

L_1:

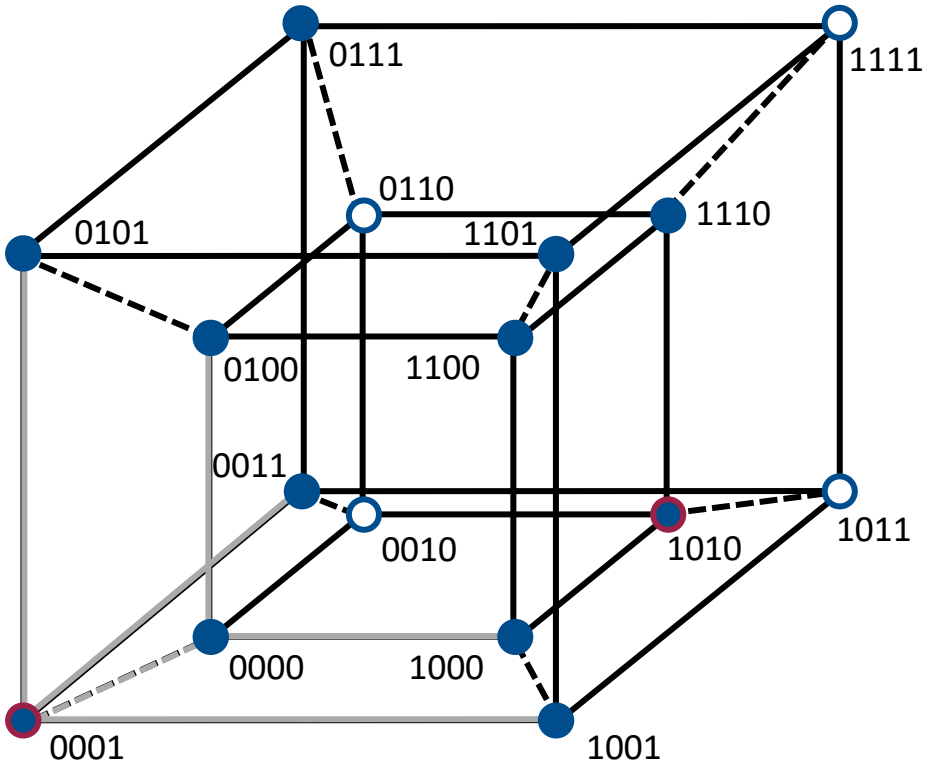
$0 \leftrightarrow 1$	$1 \leftrightarrow 2$
000−	00−1
0−00	0−01
−000	−001

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



Keine Überdeckung!

L_0:

0	1	2	3
0000	0001	0011	0111
	0100	0101	1101
	1000	1001	1110
		1010	
		1100	

L_1:

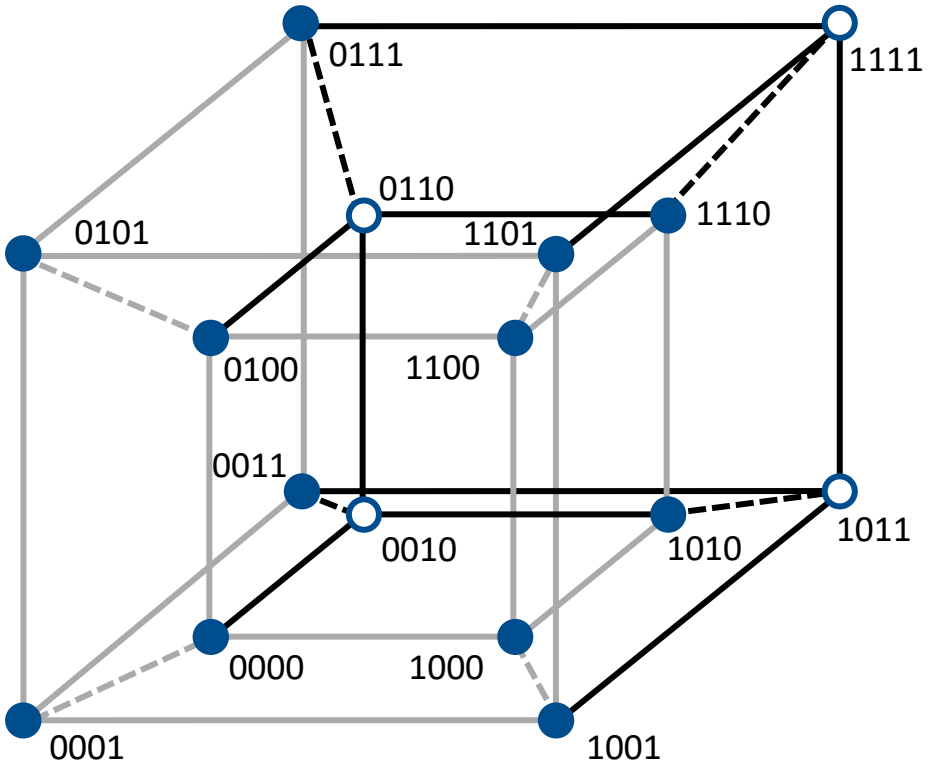
$0 \leftrightarrow 1$	$1 \leftrightarrow 2$
000−	00−1
0−00	0−01
−000	−001

Das Verfahren von Quine/McCluskey - Beispiel

Erinnerung:
0111 steht für $\overline{x_1}x_2x_3x_4$

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden

Beachte Sortierung nach McCluskey!
- Partitionierung
- Sortierung



L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

$0 \leftrightarrow 1$	$1 \leftrightarrow 2$	$2 \leftrightarrow 3$
000−	00−1	0−11
0−00	0−01	01−1
−000	−001	−101
	010−	1−01
	−100	110−
	100−	1−10
	10−0	11−0
	1−00	

Das Verfahren von Quine/McCluskey - Beispiel

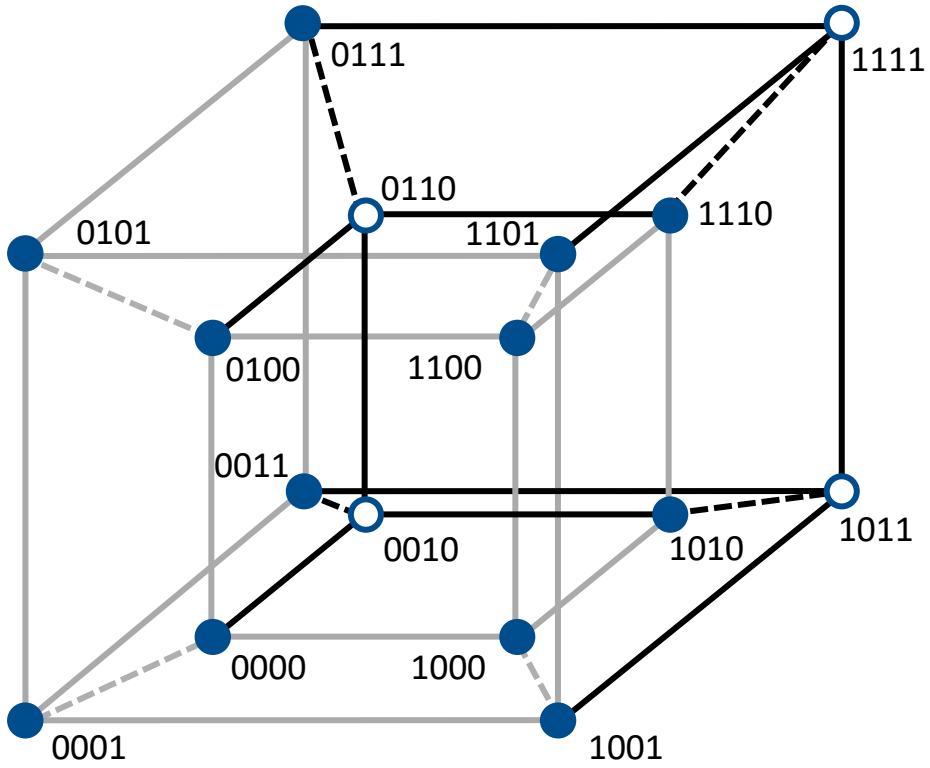
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - **Partionierung**/Sortierung nach McCluskey

L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

$L_1^{M_4}$	000–	010– 100–	110–
$L_1^{M_3}$		00–1 10–0	01–1 11–0
$L_1^{M_2}$	0–00	1–00 0–01	1–01 1–10 0–11
$L_1^{M_1}$	–000	–100 –001	–101



Das Verfahren von Quine/McCluskey - Beispiel

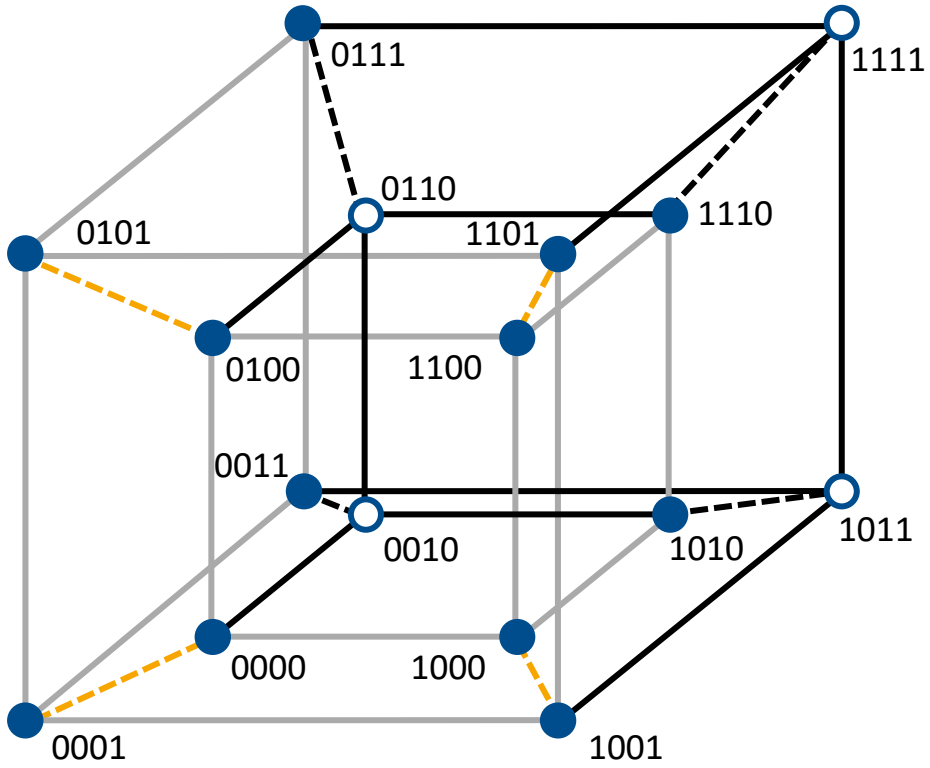
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - **Partionierung**/Sortierung nach McCluskey

L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

$L_1^{M_4}$	000–	010– 100–	110–
$L_1^{M_3}$		00–1 10–0	01–1 11–0
$L_1^{M_2}$	0–00	1–00 0–01	1–01 1–10 0–11
$L_1^{M_1}$	–000	–100 –001	–101



Das Verfahren von Quine/McCluskey - Beispiel

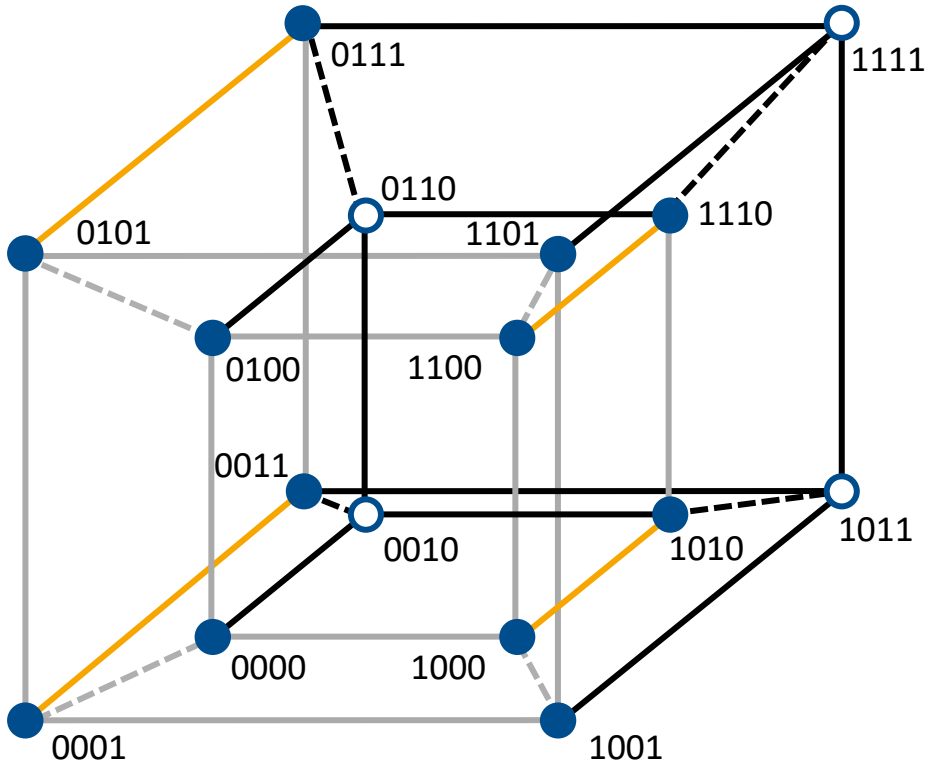
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - **Partionierung**/Sortierung nach McCluskey

L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

$L_1^{M_4}$	000–	010– 100–	110–
$L_1^{M_3}$		00–1 10–0	01–1 11–0
$L_1^{M_2}$	0–00	1–00 0–01	1–01 1–10 0–11
$L_1^{M_1}$	–000	–100 –001	–101



Das Verfahren von Quine/McCluskey - Beispiel

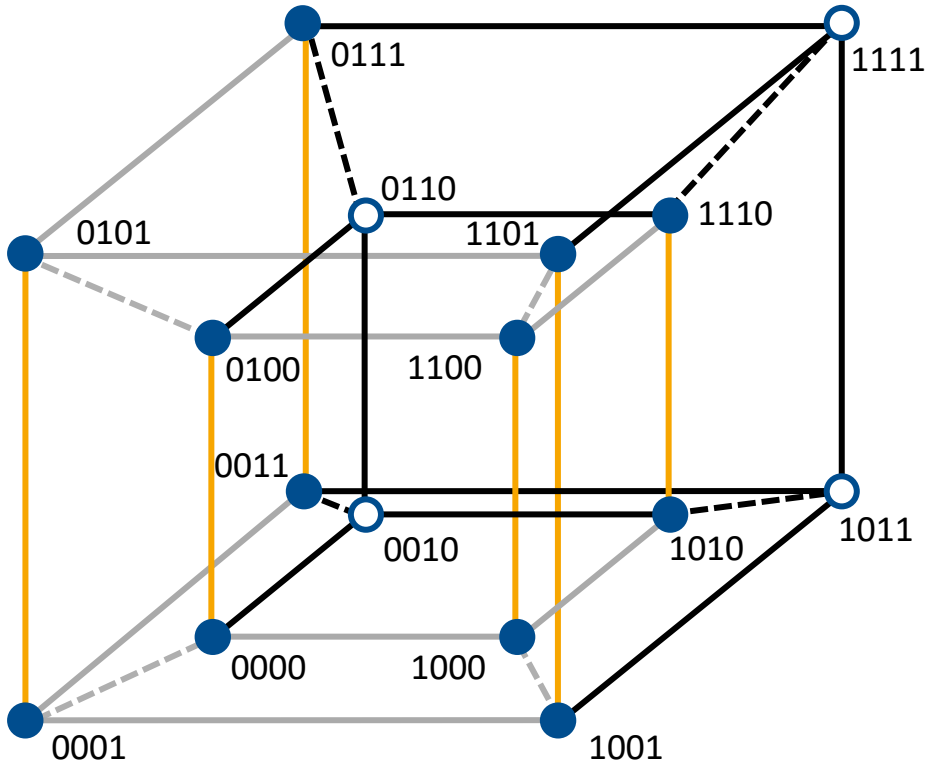
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - **Partionierung**/Sortierung nach McCluskey

L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

$L_1^{M_4}$	000–	010– 100–	110–
$L_1^{M_3}$		00–1 10–0	01–1 11–0
$L_1^{M_2}$	0–00	1–00 0–01	1–01 1–10 0–11
$L_1^{M_1}$	–000	–100 –001	–101



Das Verfahren von Quine/McCluskey - Beispiel

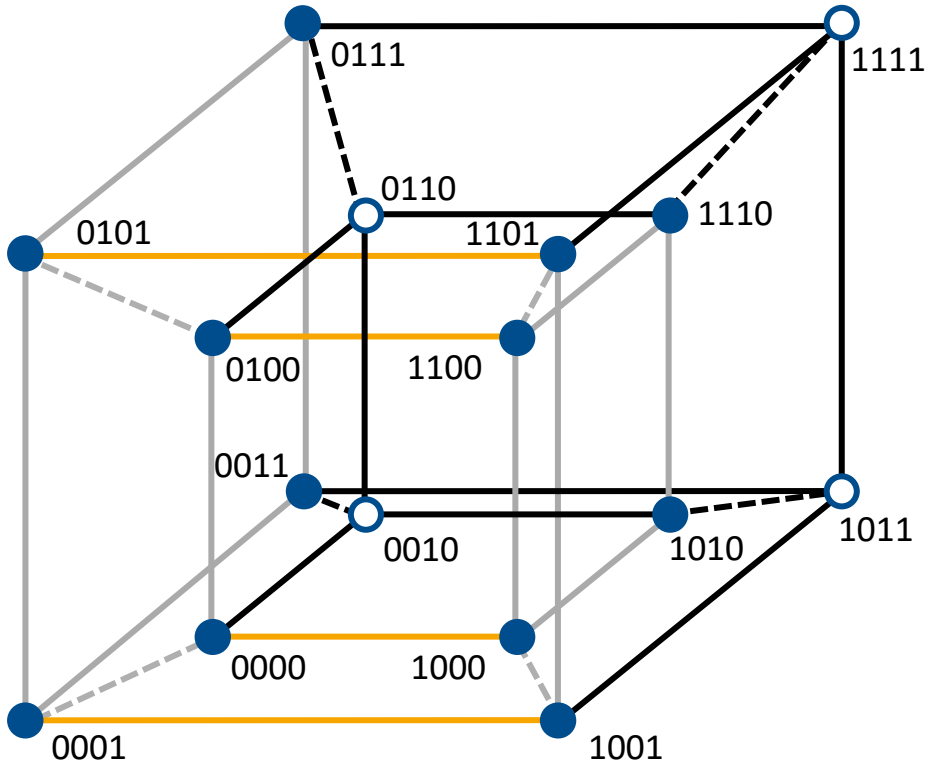
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - **Partionierung**/Sortierung nach McCluskey

L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

$L_1^{M_4}$	000–	010– 100–	110–
$L_1^{M_3}$		00–1 10–0	01–1 11–0
$L_1^{M_2}$	0–00	1–00 0–01	1–01 1–10 0–11
$L_1^{M_1}$	–000	–100 –001	–101



Das Verfahren von Quine/McCluskey - Beispiel

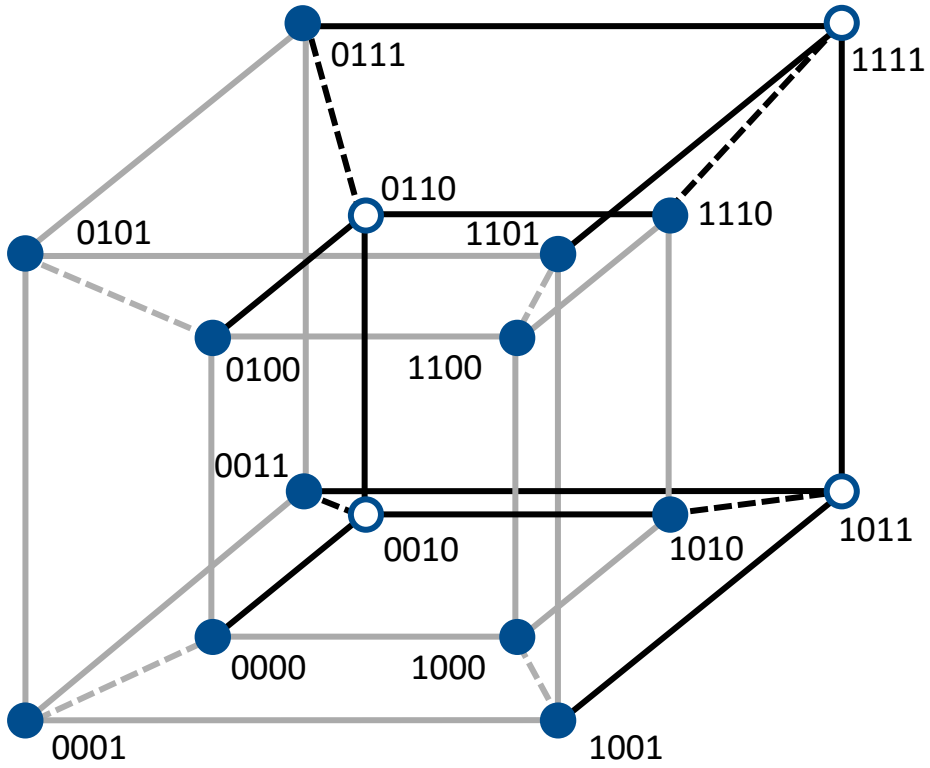
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_0 := Minterm(f)$
  - $Prim(f) := \emptyset$
  - **Partionierung**/Sortierung nach McCluskey

L_0:

0	1	2	3
0000	0001 0100 1000	0011 0101 1001 1010 1100	0111 1101 1110

L_1:

$L_1^{M_4}$	000–	010– 100–	110–
$L_1^{M_3}$		00–1 10–0	01–1 11–0
$L_1^{M_2}$	0–00	1–00 0–01	1–01 1–10 0–11
$L_1^{M_1}$	–000	–100 –001	–101



Alle Ecken überdeckt –
Primimplikanten bleiben leer

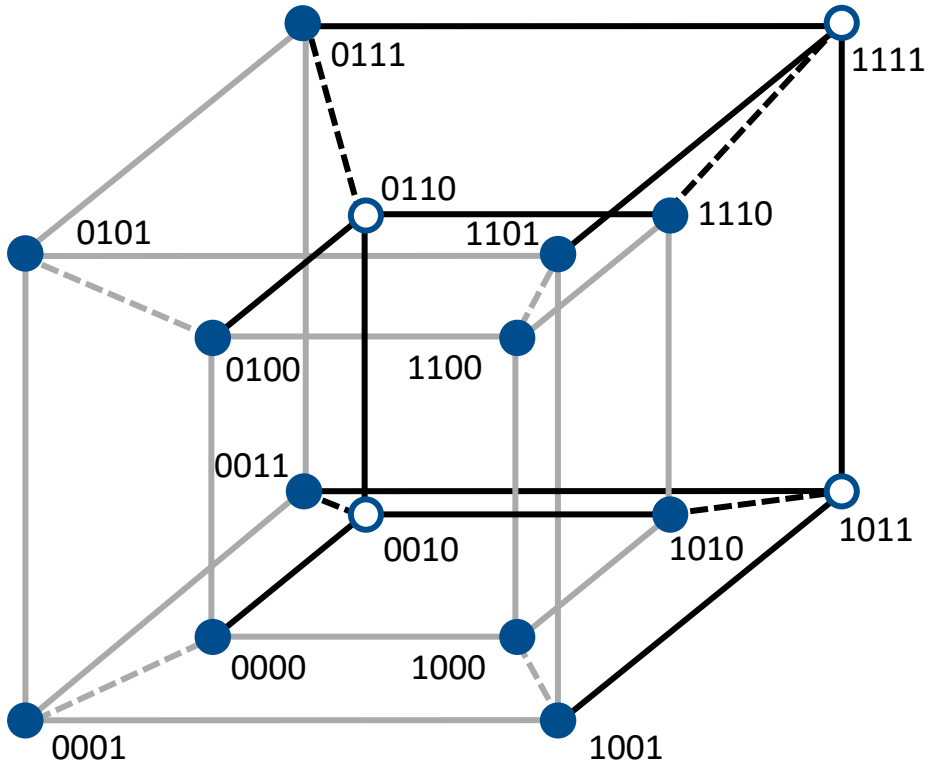
Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_1:

$L_1^{M_4}$	$000-$	$010-$ $100-$	$110-$
$L_1^{M_3}$		$00-1$ $10-0$	$01-1$ $11-0$
$L_1^{M_2}$	$0-00$	$1-00$ $0-01$	$1-01$ $1-10$ $0-11$
$L_1^{M_1}$	-000	-100 -001	-101

L_2:



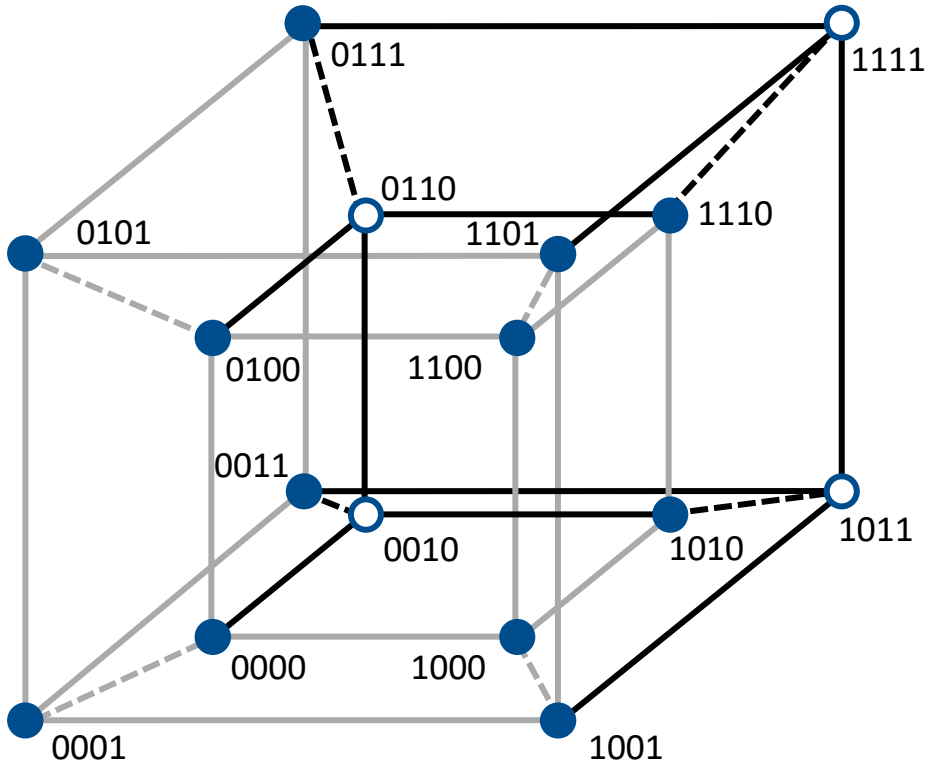
Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_1:

$L_1^{M_4}$	000−	010− 100−	110−
$L_1^{M_3}$		00−1 10−0	01−1 11−0
$L_1^{M_2}$	0−00	1−00 0−01	1−01 1−10 0−11
$L_1^{M_1}$	−000	−100 −001	−101

L_2: $\mathbf{0 \leftrightarrow 1}$



Das Verfahren von Quine/McCluskey - Beispiel

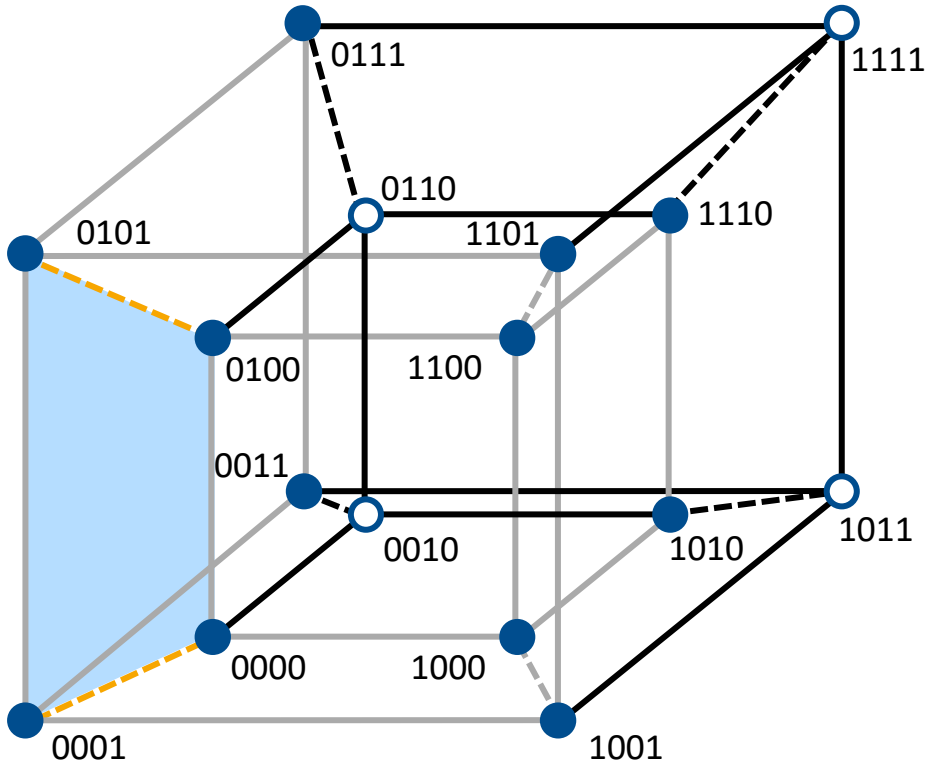
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_1:

$L_1^{M_4}$	000−	010− 100−	110−
$L_1^{M_3}$		00−1 10−0	01−1 11−0
$L_1^{M_2}$	0−00	1−00 0−01	1−01 1−10 0−11
$L_1^{M_1}$	−000	−100 −001	−101

L_2: $\mathbf{0 \leftrightarrow 1}$

0−0−



Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

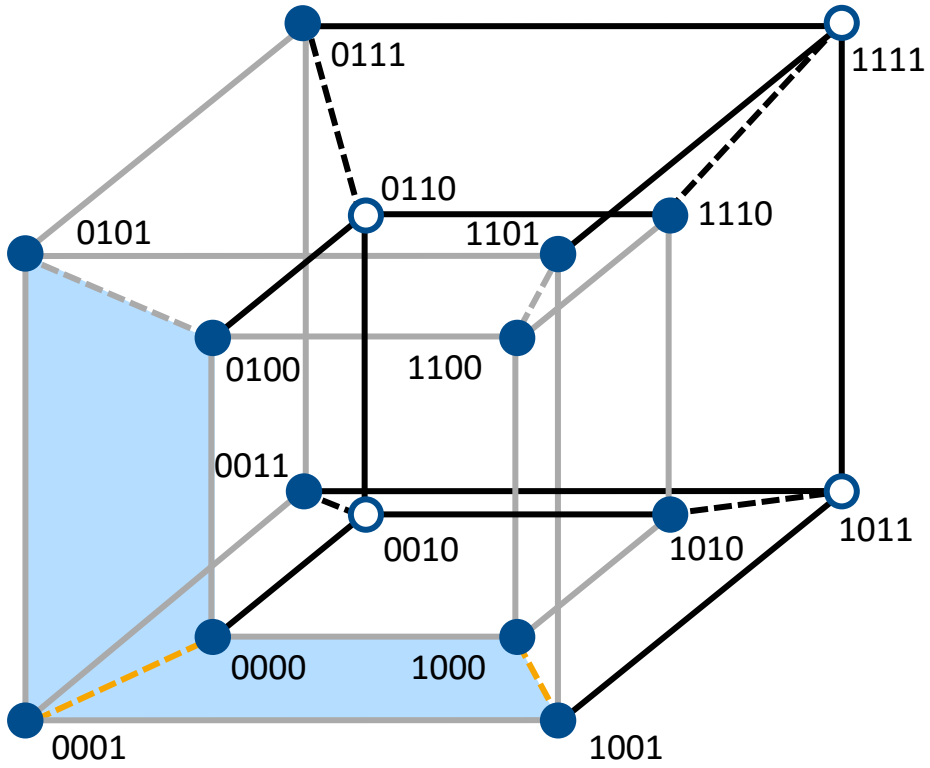
L_1:

$L_1^{M_4}$	000−	010− 100−	110−
$L_1^{M_3}$		00−1 10−0	01−1 11−0
$L_1^{M_2}$	0−00	1−00 0−01	1−01 1−10 0−11
$L_1^{M_1}$	−000	−100 −001	−101

L_2: $\mathbf{0 \leftrightarrow 1}$

0−0−

−00−



Das Verfahren von Quine/McCluskey - Beispiel

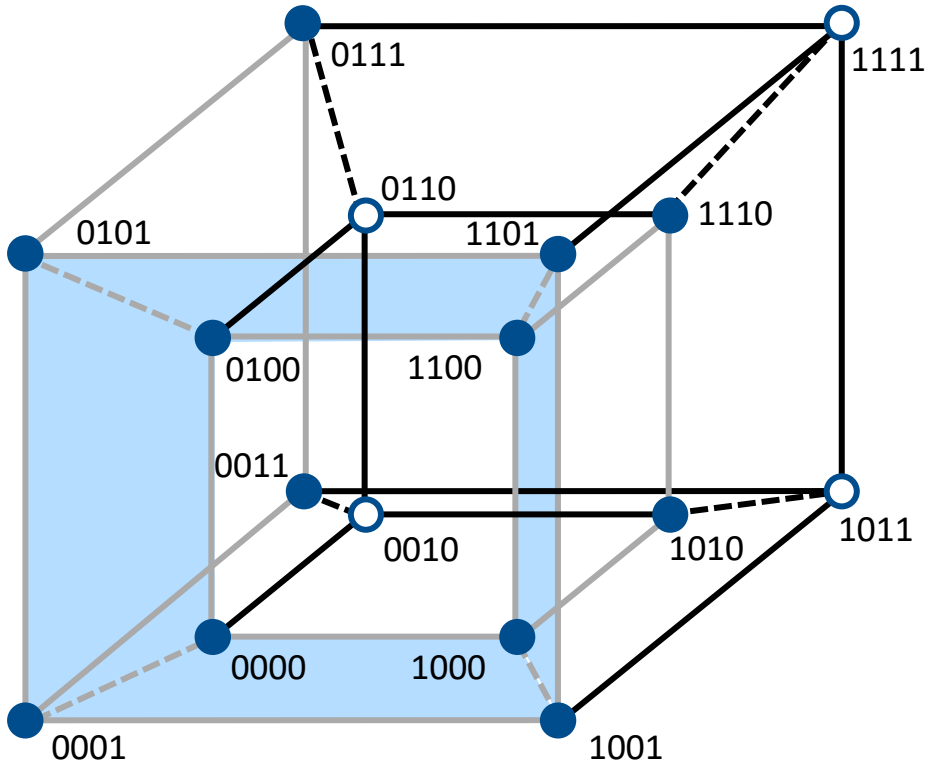
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_1:

$L_1^{M_4}$	$000-$	$010-$ $100-$	$110-$
$L_1^{M_3}$		$00-1$ $10-0$	$01-1$ $11-0$
$L_1^{M_2}$	$0-00$	$1-00$ $0-01$	$1-01$ $1-10$ $0-11$
$L_1^{M_1}$	-000	-100 -001	-101

L_2:

$\mathbf{0 \leftrightarrow 1}$	$\mathbf{1 \leftrightarrow 2}$
$0-0-$	$0-0-$
$-00-$	$1-0-$



Das Verfahren von Quine/McCluskey - Beispiel

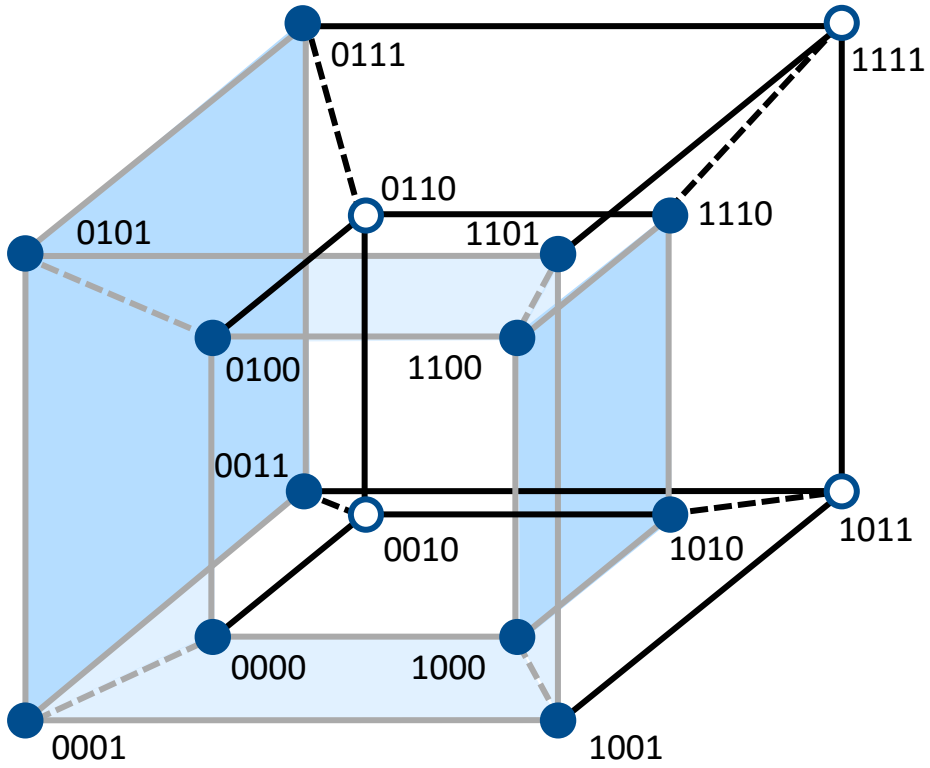
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_1:

$L_1^{M_4}$	000−	010− 100−	110−
$L_1^{M_3}$		00−1 10−0	01−1 11−0
$L_1^{M_2}$	0−00	1−00 0−01	1−01 1−10 0−11
$L_1^{M_1}$	−000	−100 −001	−101

L_2:

$\mathbf{0 \leftrightarrow 1}$	$\mathbf{1 \leftrightarrow 2}$
0−0−	0−0−
−00−	1−0−
	0−−1
	1−−0



Das Verfahren von Quine/McCluskey - Beispiel

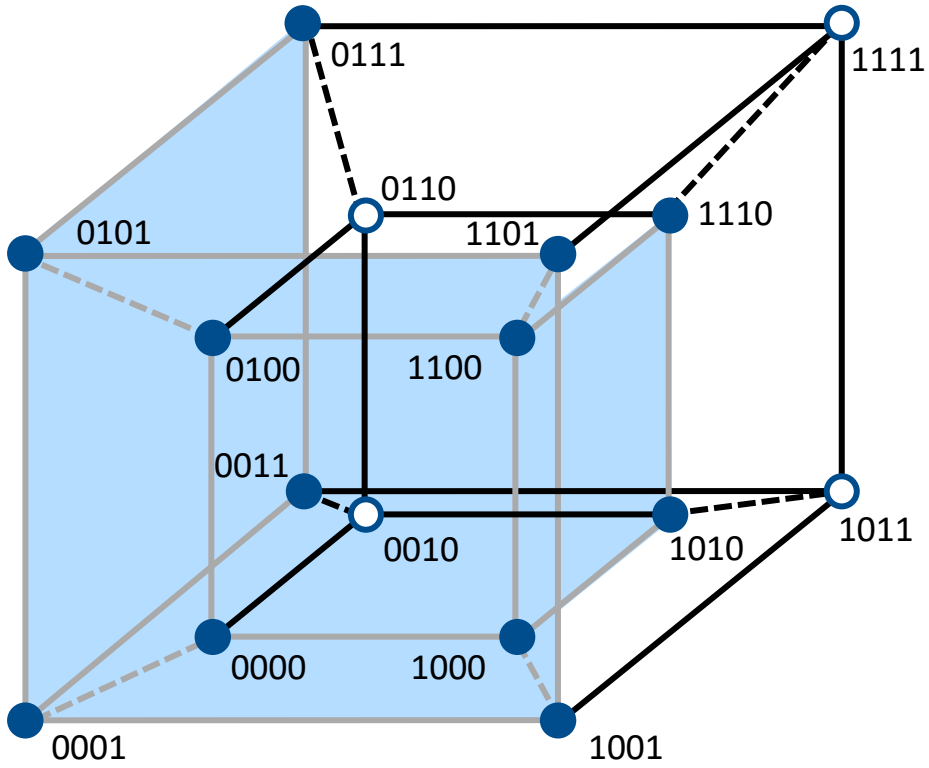
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_1:

$L_1^{M_4}$	000−	010− 100−	110−
$L_1^{M_3}$		00−1 10−0	01−1 11−0
$L_1^{M_2}$	0−00	1−00 0−01	1−01 1−10 0−11
$L_1^{M_1}$	−000	−100 −001	−101

L_2:

$0 \leftrightarrow 1$	$1 \leftrightarrow 2$
0−0− −00−	0−0− 1−0−
	0−−1 1−−0
−−00 0−0−	1−−0 0−−1 1−0− −−01



Das Verfahren von Quine/McCluskey - Beispiel

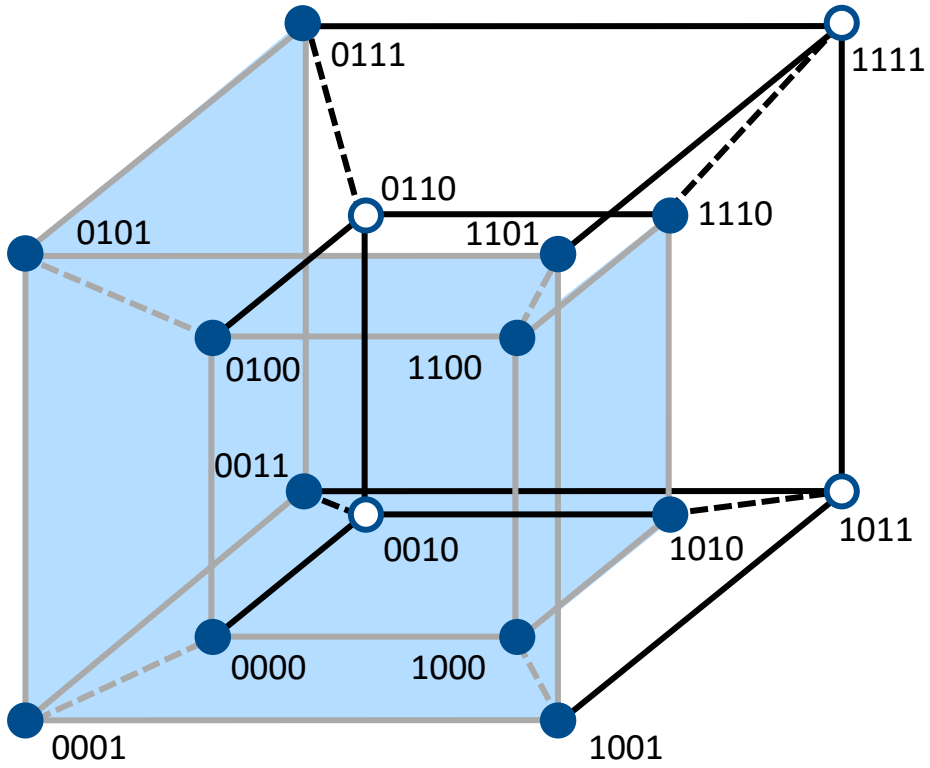
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_1:

$L_1^{M_4}$	000−	010− 100−	110−
$L_1^{M_3}$		00−1 10−0	01−1 11−0
$L_1^{M_2}$	0−00	1−00 0−01	1−01 1−10 0−11
$L_1^{M_1}$	−000	−100 −001	−101

L_2:

$0 \leftrightarrow 1$		$1 \leftrightarrow 2$	
0−0−		0−0−	
−00−		1−0−	
		0−−1	
		1−−0	
−−00	1−−0	0−−1	
0−0−	1−0−	−−01	
−−00		−10−	
−00−		−−01	



Das Verfahren von Quine/McCluskey - Beispiel

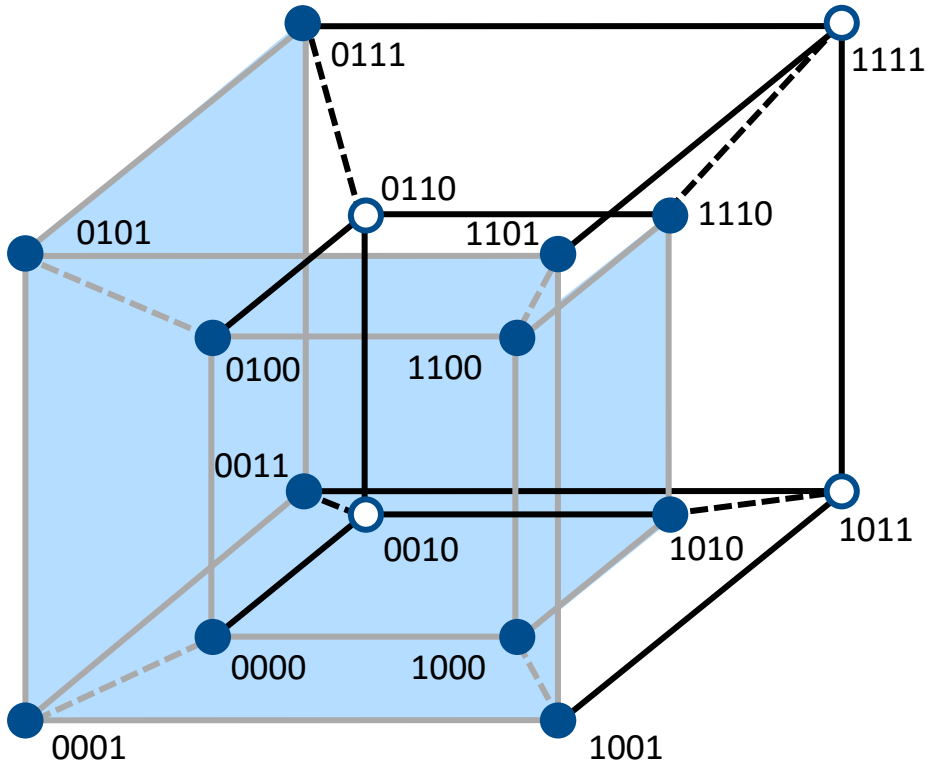
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_1 := \{000-, 010-, 100-, 110-\} \cup \{00-1, 10-0, 01-1, 11-0\} \cup \{0-00, 1-00, 0-01, 1-01, 1-10, 0-11\} \cup \{-000, -100, -001, -101\}$
  - $Prim(f) := \emptyset$
  - **Partionierung**/Sortierung nach McCluskey

L_1:

$L_1^{M_4}$	$000-$	$010-$ $100-$	$110-$
$L_1^{M_3}$		$00-1$ $10-0$	$01-1$ $11-0$
$L_1^{M_2}$	$0-00$	$1-00$ $0-01$	$1-01$ $1-10$ $0-11$
$L_1^{M_1}$	-000	-100 -001	-101

L_2:

$L_2^{M_{3,4}}$			
$L_2^{M_{2,3}}$		$0--0$ $1--0$	
$L_2^{M_{1,2}}$	$--00$	$--01$	
$L_2^{M_{2,4}}$	$0-0-$	$1-0-$	
$L_2^{M_{1,3}}$			
$L_2^{M_{1,4}}$	$-00-$	$-10-$	



Alle Kanten überdeckt –
Primimplikanten bleiben leer

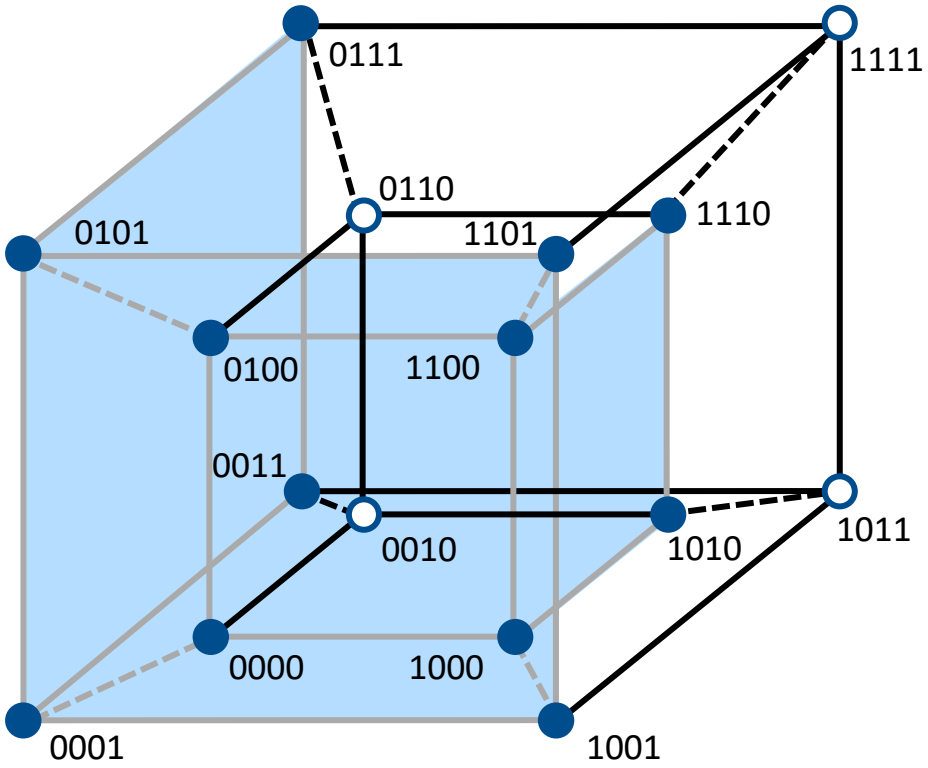
Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_2 := \{0--1, 1--0\} \cup \{--00, --01\} \cup \{0-0-, 1-0-\} \cup \{-00-, -10-\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_2:

$L_2^{M_{3,4}}$			
$L_2^{M_{2,3}}$		$0--0$ $1--0$	
$L_2^{M_{1,2}}$	$--00$	$--01$	
$L_2^{M_{2,4}}$	$0-0-$	$1-0-$	
$L_2^{M_{1,3}}$			
$L_2^{M_{1,4}}$	$-00-$	$-10-$	

L_3:



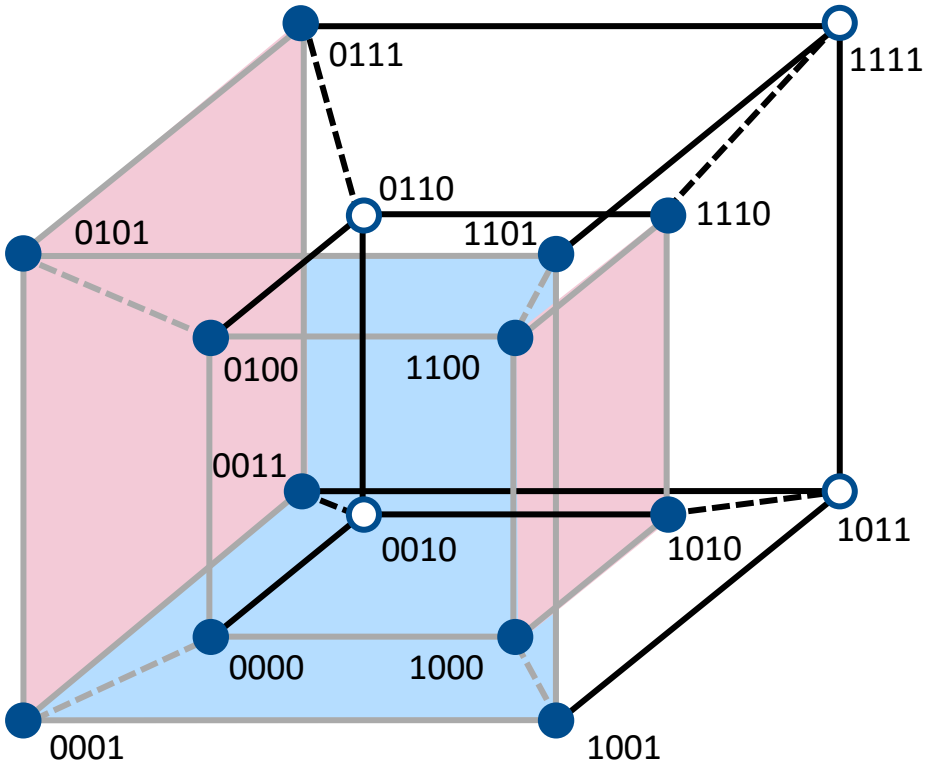
Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_2 := \{0--1, 1--0\} \cup \{--00, --01\} \cup \{0-0-, 1-0-\} \cup \{-00-, -10-\}$
  - $Prim(f) := \emptyset$
  - Überdeckungen finden (partitionsweise)

L_2:

$L_2^{M_{3,4}}$		
$L_2^{M_{2,3}}$		**0--0** **1--0**
$L_2^{M_{1,2}}$	$--00$	$--01$
$L_2^{M_{2,4}}$	$0-0-$	$1-0-$
$L_2^{M_{1,3}}$		
$L_2^{M_{1,4}}$	$-00-$	$-10-$

L_3:



Keine Überdeckung möglich

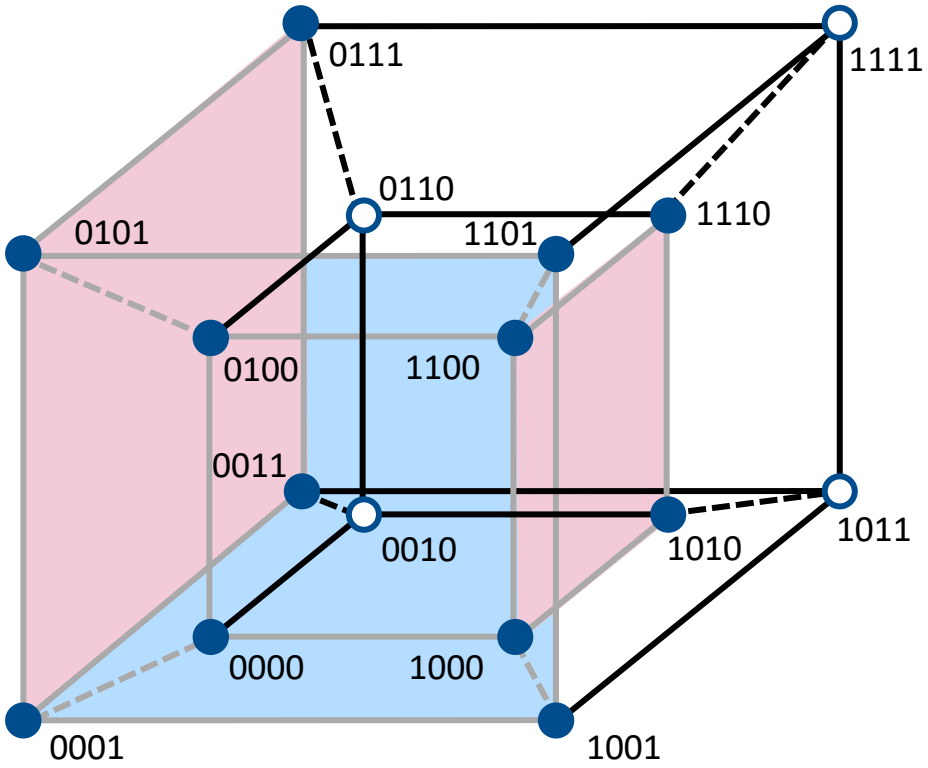
Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_2 := \{0--1, 1--0\} \cup \{--00, --01\} \cup \{0-0-, 1-0-\} \cup \{-00-, -10-\}$
  - $Prim(f) := \{\mathbf{0--1}, \mathbf{1--0}\}$
  - Überdeckungen finden (partitionsweise)

L_2:

$L_2^{M_{3,4}}$			
$L_2^{M_{2,3}}$		**0--0** **1--0**	
$L_2^{M_{1,2}}$	$--00$	$--01$	
$L_2^{M_{2,4}}$	$0-0-$	$1-0-$	
$L_2^{M_{1,3}}$			
$L_2^{M_{1,4}}$	$-00-$	$-10-$	

L_3:



Keine Überdeckung möglich – Primimplikanten!

Das Verfahren von Quine/McCluskey - Beispiel

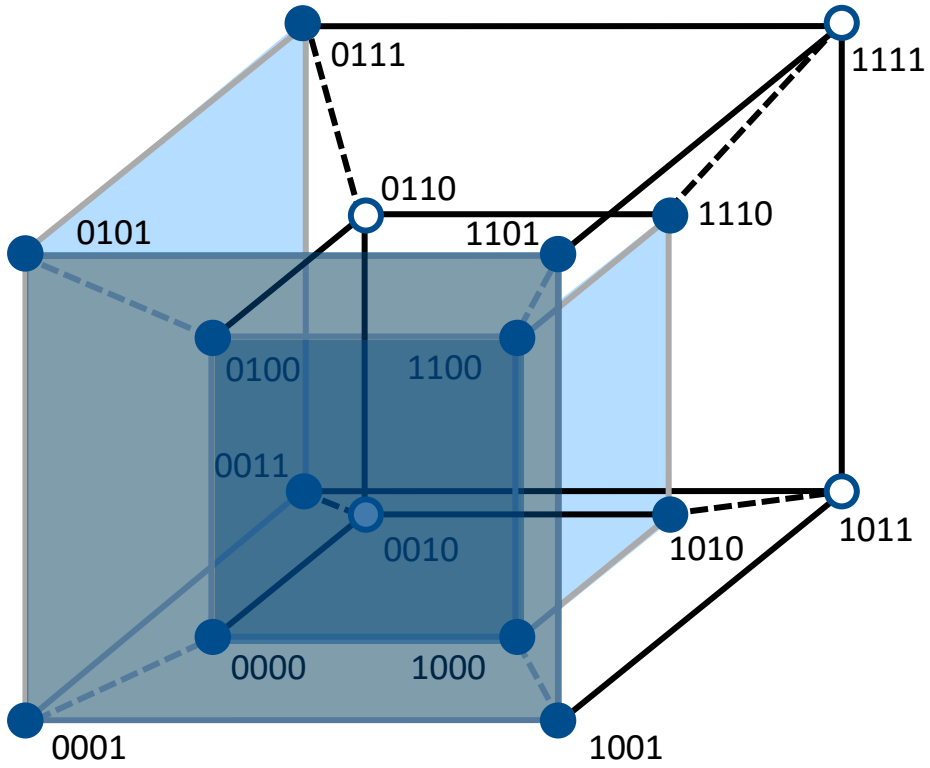
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_2 := \{0--1, 1--0\} \cup \{--00, --01\} \cup \{0-0-, 1-0-\} \cup \{-00-, -10-\}$
  - $Prim(f) := \{0--1, 1--0\}$
  - Überdeckungen finden (partitionsweise)

L_2:

$L_2^{M_{3,4}}$		
$L_2^{M_{2,3}}$		$0--0$ $1--0$
$L_2^{M_{1,2}}$	$--00$	$--01$
$L_2^{M_{2,4}}$	$0-0-$	$1-0-$
$L_2^{M_{1,3}}$		
$L_2^{M_{1,4}}$	$-00-$	$-10-$

L_3: $\mathbf{0 \leftrightarrow 1}$

$--0-$



Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_2 := \{0--1, 1--0\} \cup \{--00, --01\} \cup \{0-0-, 1-0-\} \cup \{-00-, -10-\}$
  - $Prim(f) := \{0--1, 1--0\}$
  - Überdeckungen finden (partitionsweise)

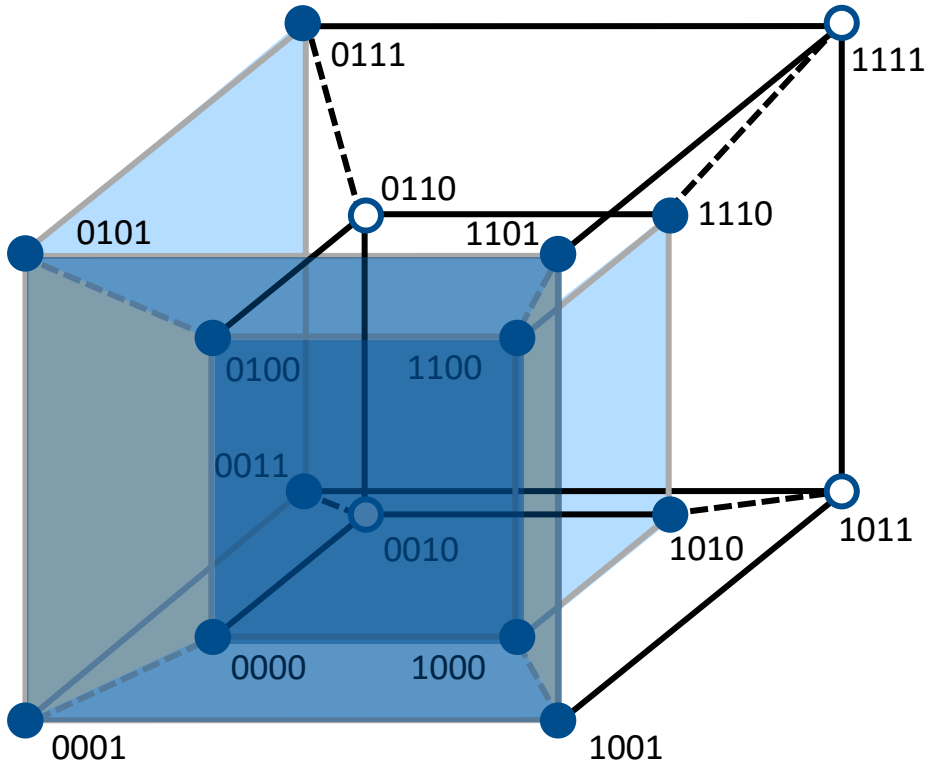
L_2:

$L_2^{M_{3,4}}$			
$L_2^{M_{2,3}}$		$0--0$ $1--0$	
$L_2^{M_{1,2}}$	$--00$	$--01$	
$L_2^{M_{2,4}}$	**$0-0-$**	**$1-0-$**	
$L_2^{M_{1,3}}$			
$L_2^{M_{1,4}}$	$-00-$	$-10-$	

L_3: $\mathbf{0 \leftrightarrow 1}$

$--0-$

$--0-$



Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_2 := \{0--1, 1--0\} \cup \{--00, --01\} \cup \{0-0-, 1-0-\} \cup \{-00-, -10-\}$
  - $Prim(f) := \{0--1, 1--0\}$
  - Überdeckungen finden (partitionsweise)

L_2:

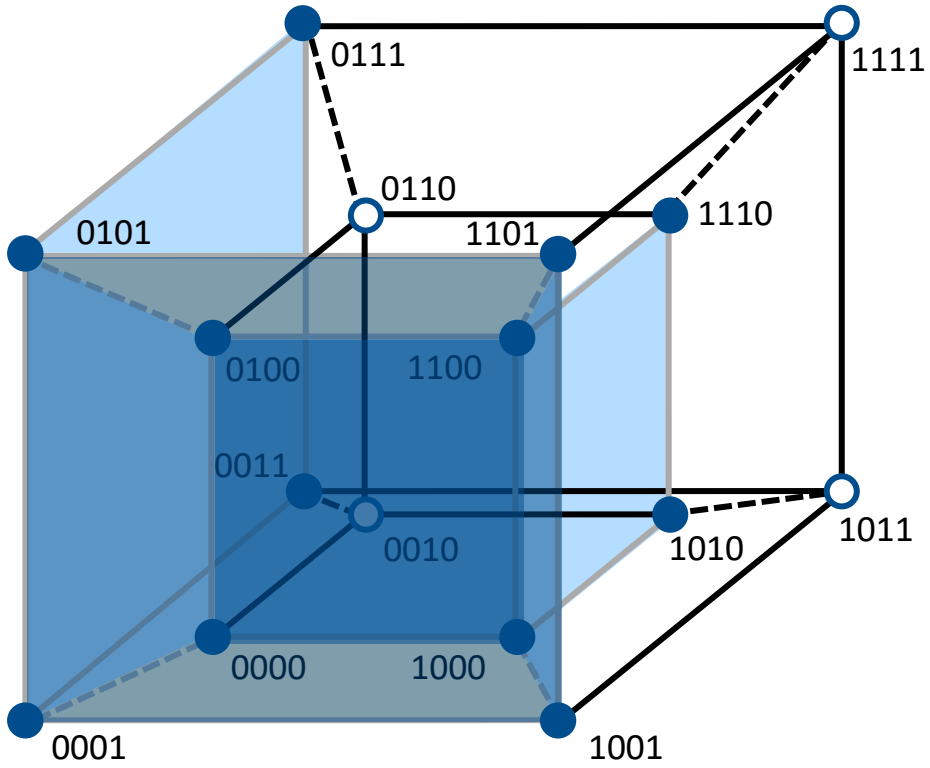
$L_2^{M_{3,4}}$			
$L_2^{M_{2,3}}$		$0--0$ $1--0$	
$L_2^{M_{1,2}}$	$--00$	$--01$	
$L_2^{M_{2,4}}$	$0-0-$	$1-0-$	
$L_2^{M_{1,3}}$			
$L_2^{M_{1,4}}$	**$-00-$**	**$-10-$**	

L_3: $\mathbf{0 \leftrightarrow 1}$

$--0-$

$--0-$

$--0-$



Das Verfahren von Quine/McCluskey - Beispiel

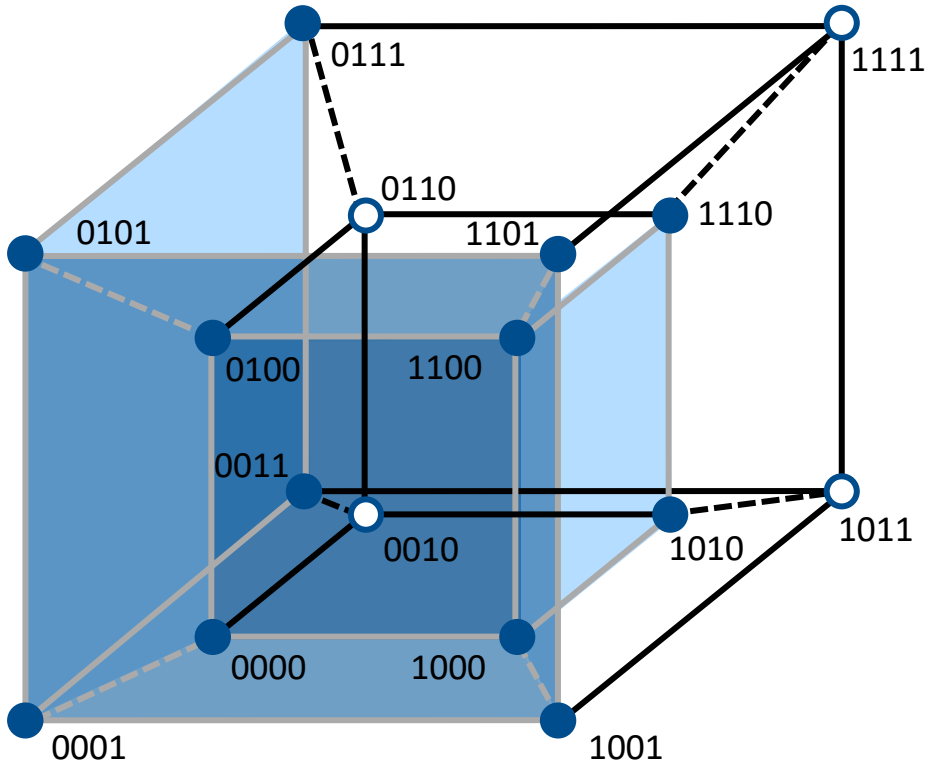
- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_2 := \{0--1, 1--0\} \cup \{--00, --01\} \cup \{0-0-, 1-0-\} \cup \{-00-, -10-\}$
  - $Prim(f) := \{0--1, 1--0\}$
  - **Partionierung**/Sortierung nach McCluskey

L_2:

$L_2^{M_{3,4}}$			
$L_2^{M_{2,3}}$		$0--0$ $1--0$	
$L_2^{M_{1,2}}$	$--00$	$--01$	
$L_2^{M_{2,4}}$	$0-0-$	$1-0-$	
$L_2^{M_{1,3}}$			
$L_2^{M_{1,4}}$	$-00-$	$-10-$	

L_3:

$L_2^{M_{2,3,4}}$	
$L_2^{M_{1,3,4}}$	
$L_2^{M_{1,2,4}}$	$--0-$
$L_2^{M_{1,2,3}}$	

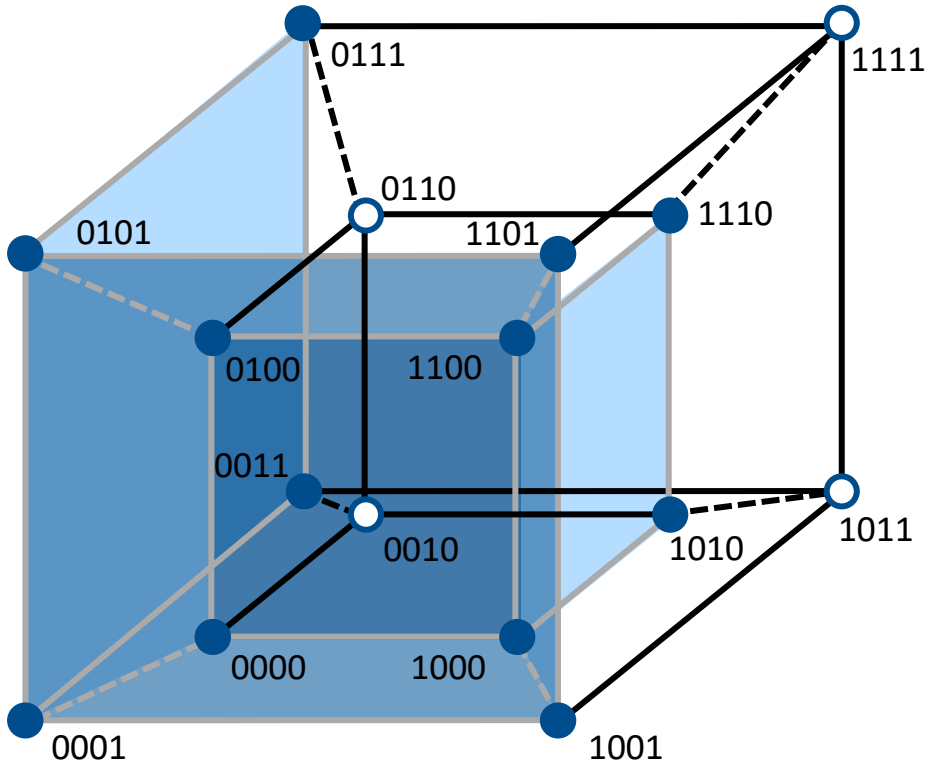


Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_3 := \{--0-\}$
  - $Prim(f) := \{0--1, 1--0\}$
  - Abbruchbedingung erfüllt

L_3:

$L_2^{M_{2,3,4}}$	
$L_2^{M_{1,3,4}}$	
$L_2^{M_{1,2,4}}$	$--0-$
$L_2^{M_{1,2,3}}$	

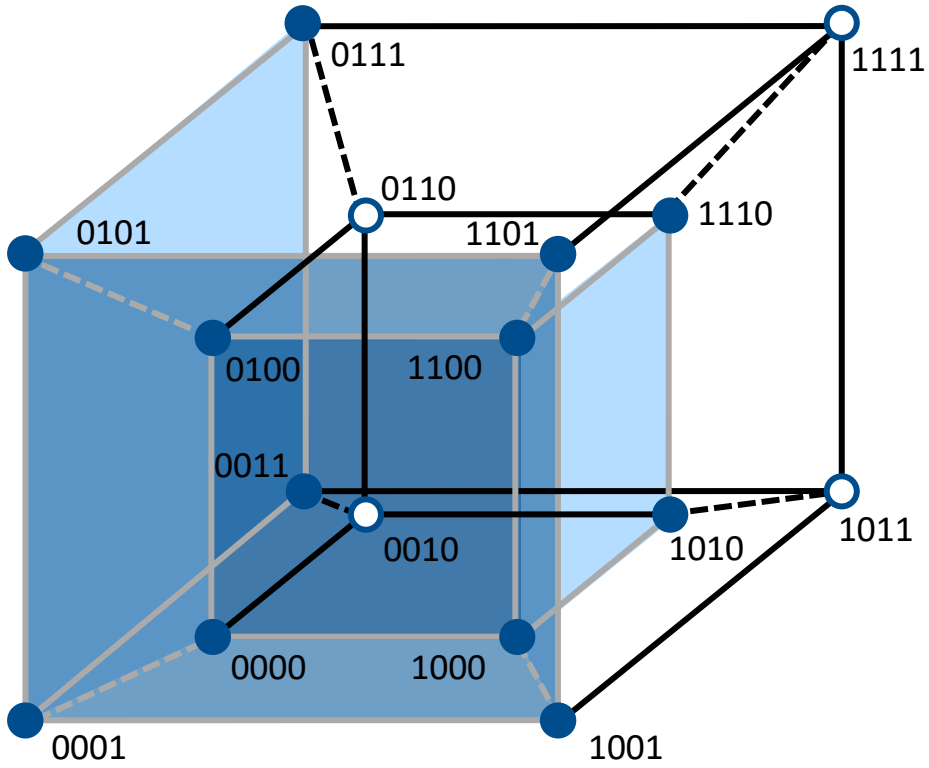


Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - $L_3 := \{--0-\}$
  - $Prim(f) := \{0--1, 1--0, \mathbf{--0-}\}$
  - Abbruchbedingung erfüllt

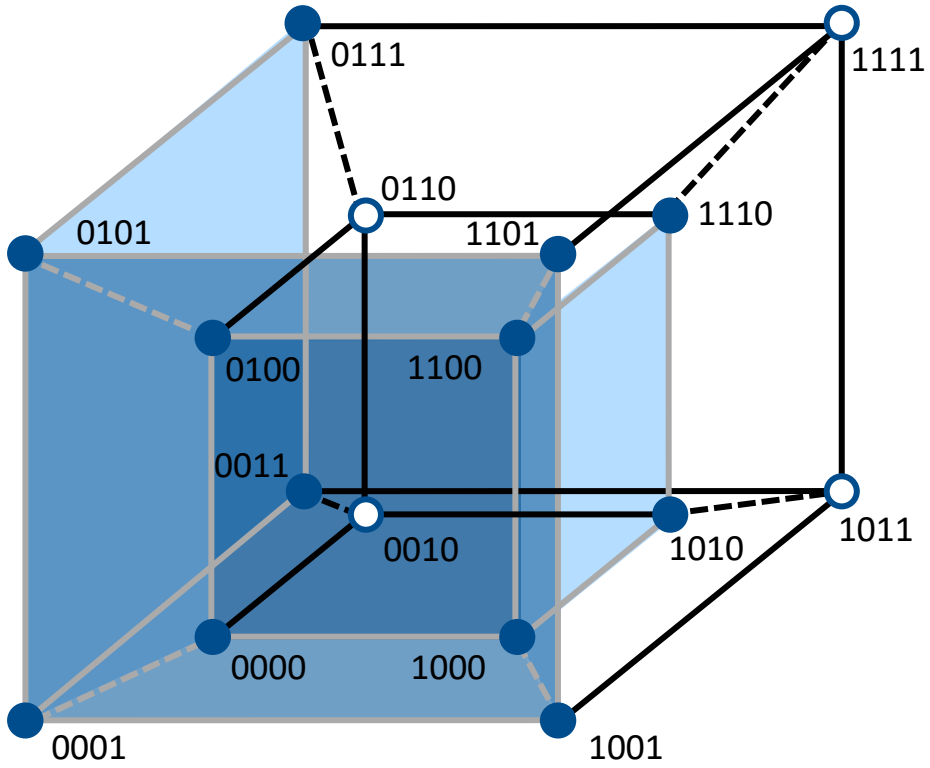
L_3:

$L_2^{M_{2,3,4}}$	
$L_2^{M_{1,3,4}}$	
$L_2^{M_{1,2,4}}$	$--0-$
$L_2^{M_{1,2,3}}$	



Das Verfahren von Quine/McCluskey - Beispiel

- Annahme: Boolesche Funktion f ist durch den Würfel gegeben
- Algorithmus:
  - Ausgabe: $Prim(f) := \{0--1, 1--0, --0-\} = \{\overline{x_1}x_4,\ x_1\overline{x_4},\ \overline{x_3}\}$



Zur Erinnerung: Das Verfahren von Quine

polynom function Quine ($f: \mathbb{B}_n \rightarrow \mathbb{B}$)

begin

$L_0 := Minterm(f);$

$i := 0;$

$Prim(f) := \emptyset;$

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

loop

$L_{i+1} := \{m | \exists i \in \{1, \ldots, n\}: \{mx_i, m\overline{x_i}\} \subset L_i\};$

$Prim(f) := Prim(f) \cup \{m | m \in L_i \; und \; m \; wird \; von \; keinem \; q \in L_{i+1} \; überdeckt\}$

$i := i + 1;$

end loop

return $Prim(f) \cup L_i$

end

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

Korrektheit von Quine-McCluskey

Theorem:
Für alle $i = 0,1, \dots, n$ gilt:
- L_i enthält nur Monome mit $n - i$ Literalen
- L_i enthält genau die Implikanten von f mit $n - i$ Literalen
- Nach Iteration i enthält $Prim(f)$ genau die Primimplikanten von f mit $n - i$ Literalen

Beweis:
Induktion über i.

Abbruchbedingungen:
- $L_i = \emptyset$ bedeutet, dass keine Implikanten bei der „Partnersuche“ entstanden sind, d.h. L_{i+1} ist ganz in $Prim(f)$aufgegangen.
- $i = n$ bedeutet, dass L_n berechnet wurde, es gilt dann $L_n = \emptyset$ oder $L_n = \{1\}$. Letzteres bedeutet f ist die „Eins“-Funktion und $Prim(f) = \{1\}$.

Kosten des Verfahrens (1)

Lemma:
Es gibt 3^n verschiedene Monome in n Variablen.

Beweis:
Für jedes Monom m und jede der n Variablen x liegt genau eine der drei folgenden Situationen vor:

- m enthält weder das positive noch das negative Literal von x
- m enthält das positive Literal x
- m enthält das negative Literal $\overline{x}$

Jedes Monom ist durch diese Beschreibung eindeutig bestimmt.

Kosten des Verfahrens (2)

Lemma:
Das Verfahren von Quine/McCluskey macht höchstens $2n!\,3^n$ Monomvergleiche.

Beweis:
Jeder Block besteht aus $\leq n!$ vielen Monomen (gleiche Anzahl von positiven Literalen nach Definition eines Blockes). Ein Monom wird mit den Implikanten der beiden benachbarten Blöcke verglichen. Dann folgt mit dem vorherigen Lemma die Behauptung.

Korollar:
Das Verfahren von Quine/McCluskey benötigt höchstens $2n!\,3^n n$ Bitvergleiche.

Beweis:
Jeder Vergleich zwischen zwei Monomen kostet $\leq n$ Bitvergleiche. Mit dem vorherigen Lemma folgt die Behauptung.

Skizze: Implementierung und Beschleunigung der Berechnung

- Reserviere im Hauptspeicher einen Bitvektor `IMPLIKANT` der Länge 3^n.
- Initialisiere den Bitvektor auf `(0,…,0)`
- Interpretiere im Folgenden jedes Monom m als Folge m der Länge n :
  - Enthält m das Literal $\overline{x_i}$, so setze `m[i]=0`
  - Enthält m das Literal x_i, so setze `m[i]=1`
  - Enthält m weder x_i noch $\overline{x_i}$, so setze `m[i]=2`
- Identifiziere m mit der von ihm dargestellten ternären Zahl $\sum m[i] \cdot 3^i$
- Setze *IMPLIKANT[m]*=1, falls Monom m als Implikant gefunden ist.

Beobachtung

- Ein Monom m kann nur zu einem Monom benachbart sein, welches durch Kippen eines (gültigen) Bits entsteht. Dies sind höchstens n Stück.
- Jeden dieser Nachbarn erhält man durch „entsprechende“ Multiplikationen und Additionen.

Komplexität des Verfahrens von Quine/McCluskey

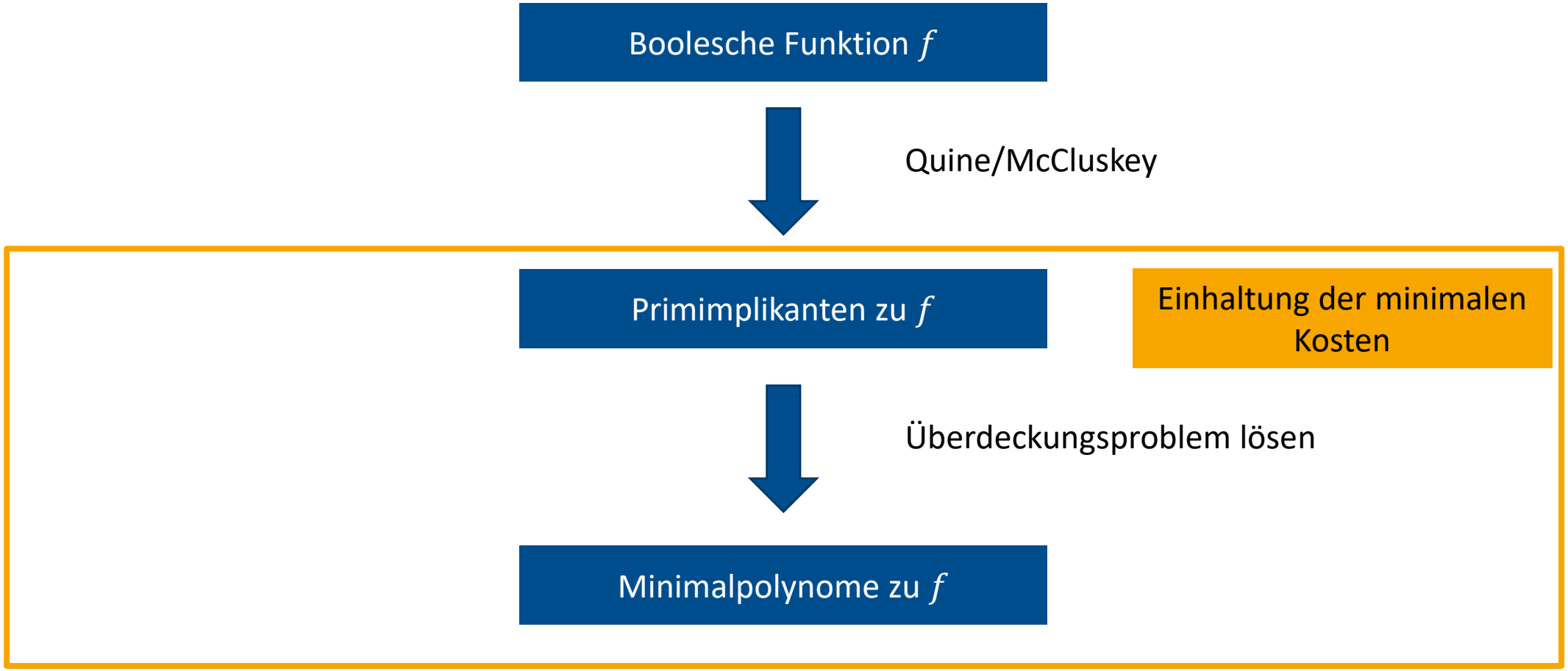
Satz (Komplexität des Verfahrens von Quine/McCluskey):
Die Laufzeit des Verfahrens liegt in $\mathcal{O}(n^2 \cdot 3^n \cdot \log(n))$ beziehungsweise in $\mathcal{O}\left(\log^2 N \cdot N^{\log 3} \cdot \log\log N\right)$, wobei $N = 2^n$ die Größe der Funktionstabelle ist.

Beweis:
Jedes der 3^n Monome wird mit höchstens n anderen Monomen verglichen.

Verwendet man schnelle Multiplikations- und Additionsschaltungen (siehe Kapitel 11), benötigt jeder Vergleich (die Berechnung des „Nachbarn“) Laufzeit $\mathcal{O}(n \cdot \log n)$. Dies ergibt die erste Behauptung.

Die zweite Teilbehauptung folgt durch Einsetzen und Nachrechnen.

Algorithmus zur Berechnung eines Minimalpolynoms



Das Matrix-Überdeckungsproblem

Gegeben:

Die Menge der Primimplikanten $Prim(f)$ einer Booleschen Funktion $f \in \mathcal{B}_n$ und die ON-Menge $ON(f) \subseteq \mathbb{B}^n$ der Booleschen Funktion f.

Gesucht:

$M \subseteq Prim(f)$, die kostenminimal ist und deren Element-Disjunktion ein Polynom von f ist.

Zur Erinnerung:

Kostenminimal bedeutet, dass $cost(p) \leq cost(p')$ mit $p = \sum_{m_i \in M} m_i$ und $p' = \sum_{m_i \in M'} m_i, M' \subseteq Prim(f)$ und $M \neq M'$

Das Matrix-Überdeckungsproblem – die Primimplikantentafel

Definition (Primimplikantentafel):
Sei $PIT(f)$ eine Boolesche Matrix mit Dimension $|Prim(f)| \times |ON(f)|$. Die Einträge der Matrix seien wie folgt definiert: $pit(f)_{ij} = 1 \Leftrightarrow \psi(m_i^p)(\alpha_j) = 1$, wobei $m_i^p \in Prim(f)$ und $\alpha_j \in ON(f)$. Diese Matrix heißt Primimplikatentafel.

Interpretation:

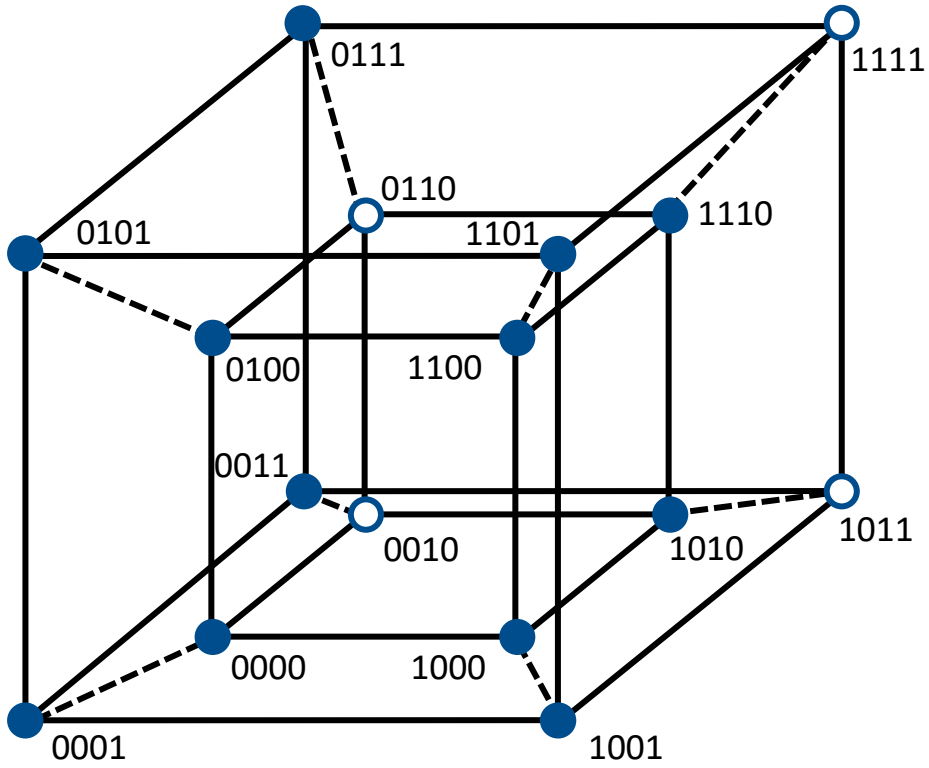
- Zeilen: Primimplikanten von f
- Spalten: Minterme von f, ausgedrückt als Minterm $m(\alpha)$ oder Belegung α
- Eintrag an der Stelle ij:
  - Der Minterm $m(\alpha_j)$ wird vom Primimplikanten m_i^p überdeckt
  - Im Würfel: Die Ecke, die $m(\alpha_j)$ entspricht, ist eine Ecke des Teilwürfels m_i^p

Gesucht:
Eine Teilmenge $M^* \subseteq Prim(f)$ mit:

- Jede Spalte (Minterm) ist von einem Primimplikanten überdeckt, d.h. $\forall \alpha_j \in ON(f), \exists m_i^p \in Prim(f): pit(f)_{ij} = 1$
- M^* ist kostenminimal

Die Primimplikantentafel – Beispiel

- Annahme: $Prim(f) = \{\overline{x_1}x_4, x_1\overline{x_4}, \overline{x_3}\}$

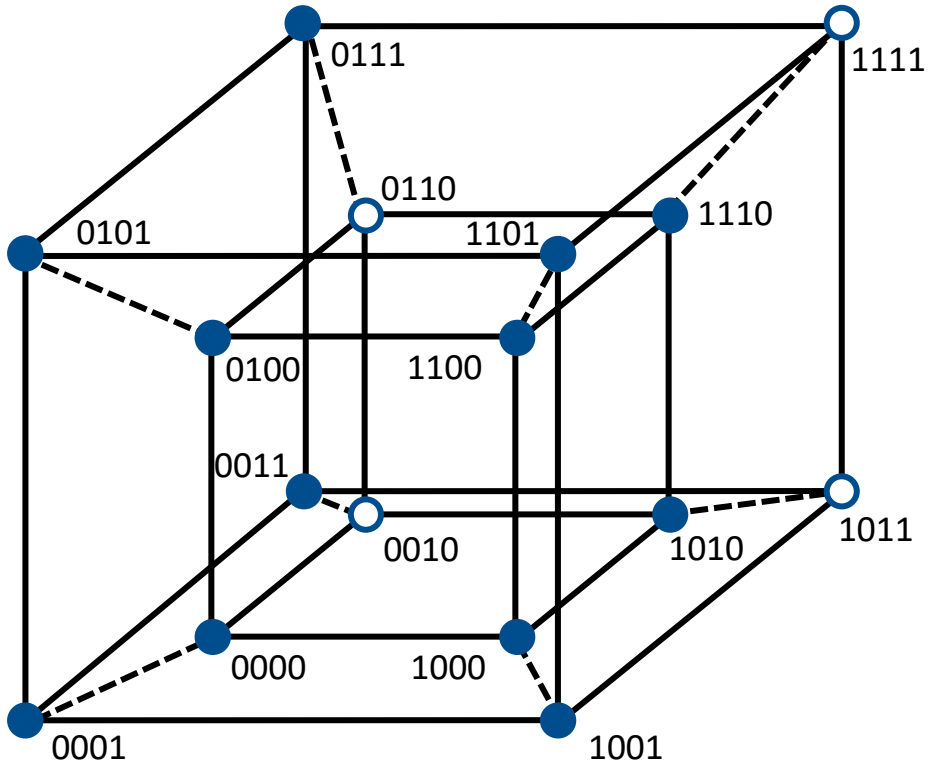


Die Primimplikantentafel – Beispiel

- Annahme: $Prim(f) = \{\overline{x_1}x_4, x_1\overline{x_4}, \overline{x_3}\}$
- Aufbau $PIT(f)$:
  - Spalten entsprechen Mintermen

	0	1	3	4	5	7	8	9	10	12	13	14

Hinweis:
Für Minterme wird die Dezimalrepräsentation verwendet

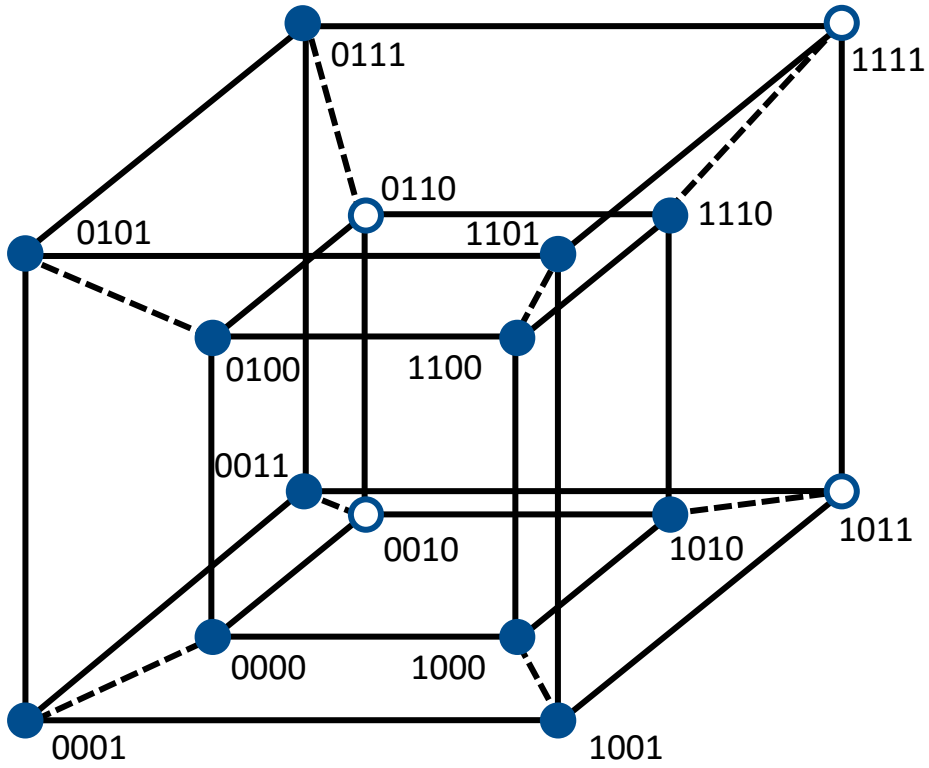


Die Primimplikantentafel – Beispiel

- Annahme: $Prim(f) = \{\overline{x_1}x_4, x_1\overline{x_4}, \overline{x_3}\}$
- Aufbau $PIT(f)$:
  - Spalten entsprechen Mintermen
  - Zeilen entsprechen Primimplikanten

	0	1	3	4	5	7	8	9	10	12	13	14
$\overline{x_1}x_4$												
$x_1\overline{x_4}$												
$\overline{x_3}$												

Hinweis:
Für Minterme wird die Dezimalrepräsentation verwendet



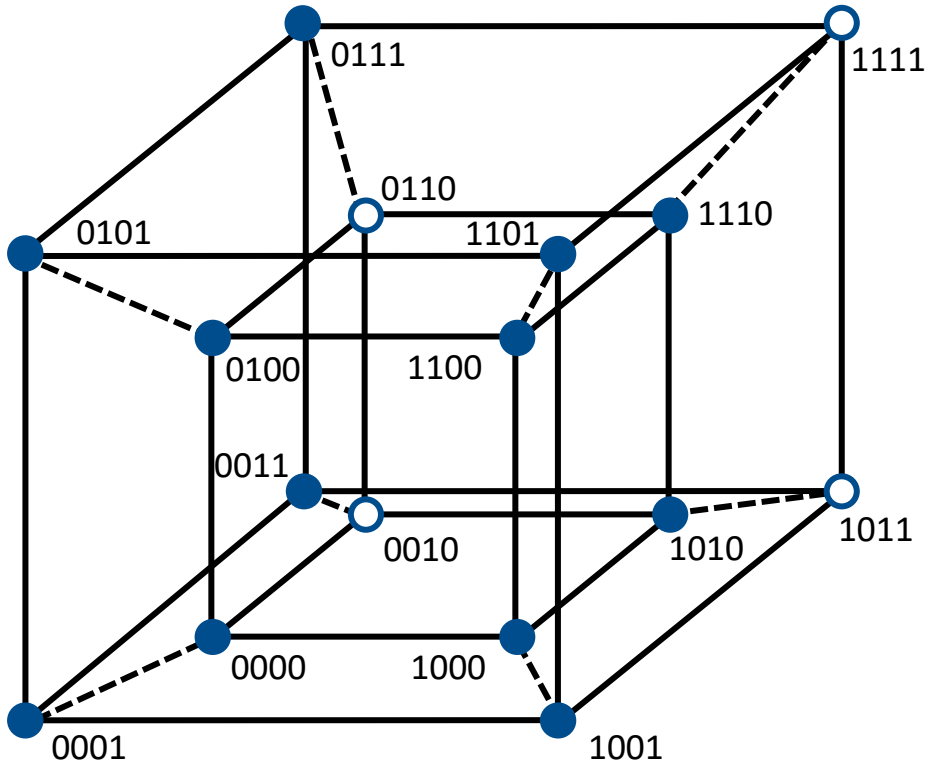
Die Primimplikantentafel – Beispiel

- Annahme: $Prim(f) = \{\overline{x_1}x_4, x_1\overline{x_4}, \overline{x_3}\}$
- Aufbau $PIT(f)$:
  - Spalten entsprechen Mintermen
  - Zeilen entsprechen Primimplikanten
  - Bei Überdeckung: 1 gesetzt

	0	1	3	4	5	7	8	9	10	12	13	14
$\overline{x_1}x_4$		1	**1**		1	1						
$x_1\overline{x_4}$							1		1	1		**1**
$\overline{x_3}$	1	1		1	1		1	**1**		1	1	

Alle Primimplikanten sind wesentlich

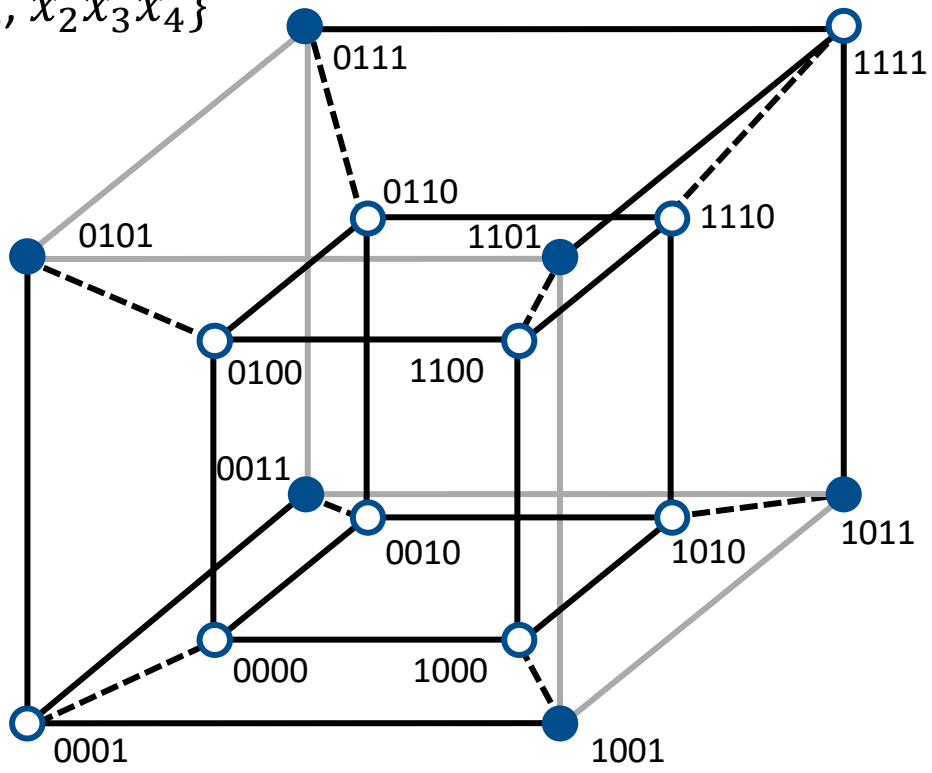
Hinweis:
Für Minterme wird die Dezimalrepräsentation verwendet



Die Primimplikantentafel – Beispiel 2

- Annahme: $Prim(f) = \{\overline{x_1}x_2x_4, \overline{x_1}x_3x_4, \overline{x_2}x_3x_4, x_1\overline{x_2}x_4, x_1\overline{x_3}x_4, x_2\overline{x_3}x_4\}$
- Aufbau $PIT(f)$:
  - Spalten entsprechen Mintermen
  - Zeilen entsprechen Primimplikanten

	3	5	7	9	11	13
$\overline{x_1}x_2x_4$		1	1			
$\overline{x_1}x_3x_4$	1		1			
$\overline{x_2}x_3x_4$	1				1	
$x_1\overline{x_2}x_4$				1	1	
$x_1\overline{x_3}x_4$				1		1
$x_2\overline{x_3}x_4$		1				1



Kein Primimplikant ist wesentlich

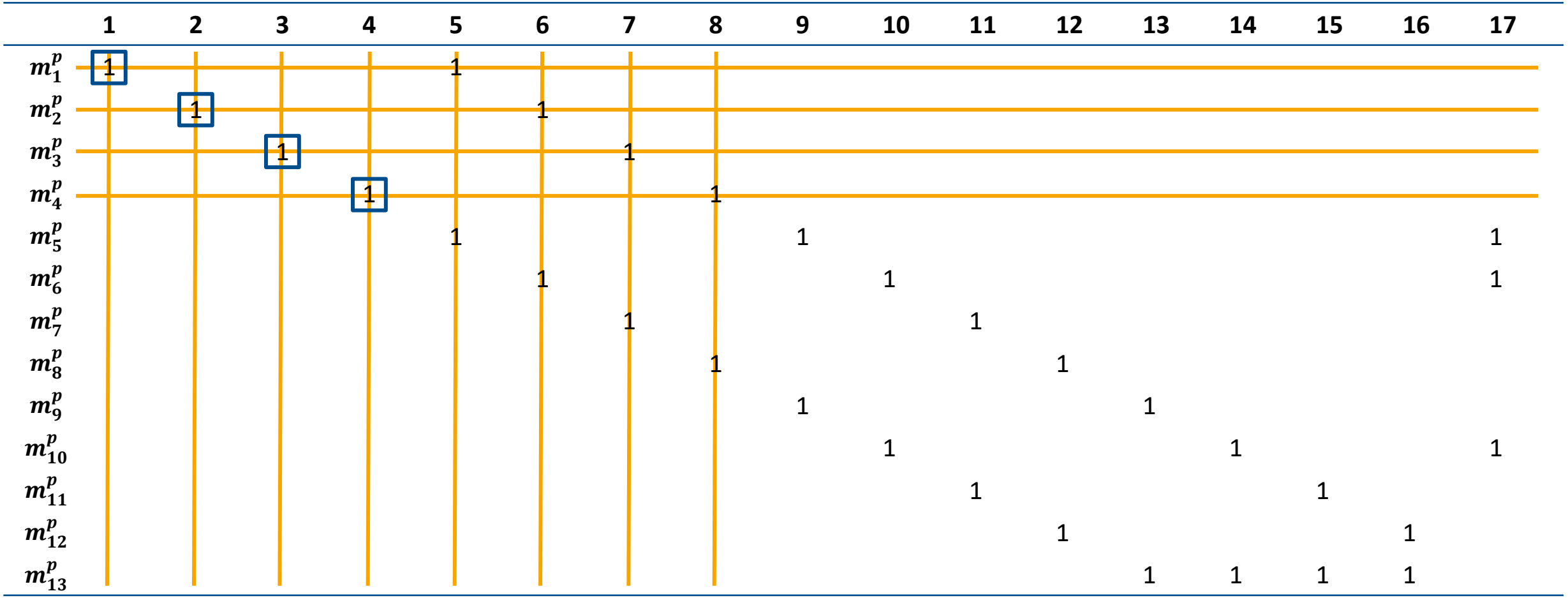
Erste Reduktionsregel - Definition

Definition (wesentlicher Primimplikant):
Ein Primimplikant m_i^p von f heißt wesentlich, wenn es einen Minterm $m(\alpha_j)$ von f gibt, der nur von diesem Primimplikanten überdeckt wird, d.h. $pit(f)_{ij} = 1$ und $pit(f)_{kj} = 0 \; \forall \, m_k^p \in Prim(f), k \neq i$.

Lemma:
Jedes Minimalpolynom von f enthält alle wesentlichen Primimplikanten von f.

Erste Reduktionsregel:
Entferne aus der Primimplikantentafel alle wesentlichen Primimplikanten und die Minterme, die von ihnen überdeckt werden und füge die wesentlichen Primimplikanten einer Menge $M^* \subseteq Prim(f)$ hinzu.

Erste Reduktionsregel - Beispiel



Erste Reduktionsregel - Beispiel

- Primimplikantentafel nach Löschen der wesentlichen Implikanten
- Kein Primimplikant ist mehr wesentlich

	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
m_5^p	1								1
m_6^p		1							1
m_7^p			1						
m_8^p				1					
m_9^p	1				1				
m_{10}^p		1				1			1
m_{11}^p			1				1		
m_{12}^p				1				1	
m_{13}^p					1	1	1	1	

Zweite Reduktionsregel - Definition

Definition (dominante Spalten/Minterme):
Sei $A \in \mathbb{B}^{n \times m}$ eine Boolesche Matrix. Die Spalte j der Matrix A dominiert die Spalte k der Matrix A, wenn $a_{ij} \geq a_{ik}, \forall\, i \in \{1, \dots, n\}$.

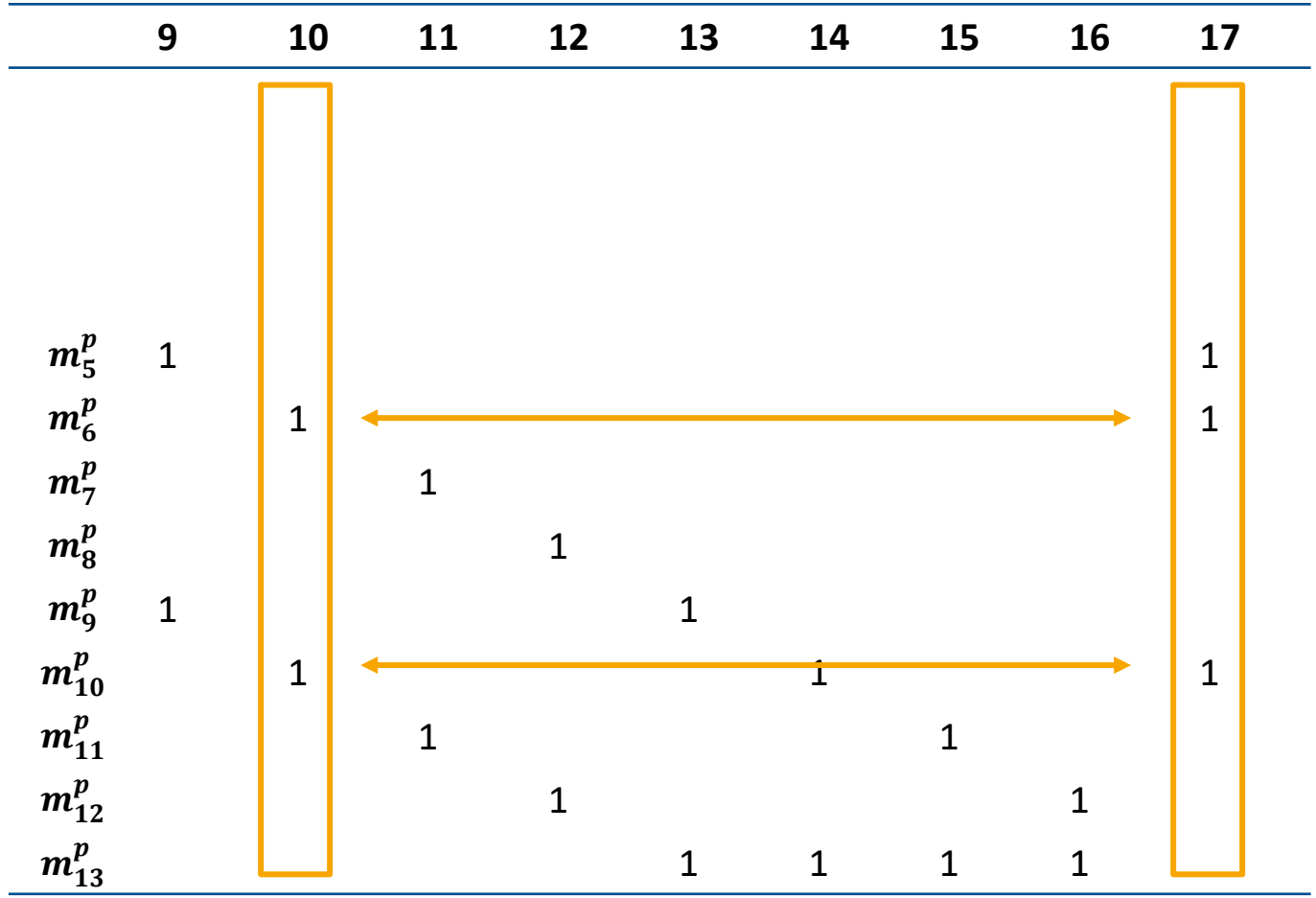
Nutzen für unser Problem:

Dominiert ein Minterm $m(\alpha_j)$ von f einen anderen Minterm $m(\alpha_k)$, so braucht $m(\alpha_j)$ nicht mehr weiter betrachtet zu werden, da $m(\alpha_k)$ auf jeden Fall überdeckt werden muss und hierdurch auch der Minterm $m(\alpha_j)$ überdeckt wird. Jeder noch vorhandene Primimplikant m_i^p, der $m(\alpha_k)$ überdeckt, überdeckt auch $m(\alpha_j)$.

Zweite Reduktionsregel:
Entferne aus der Primimplikantentafel $PIT(f)$ alle Minterme, die einen anderen Minterm in $PIT(f)$ dominieren.

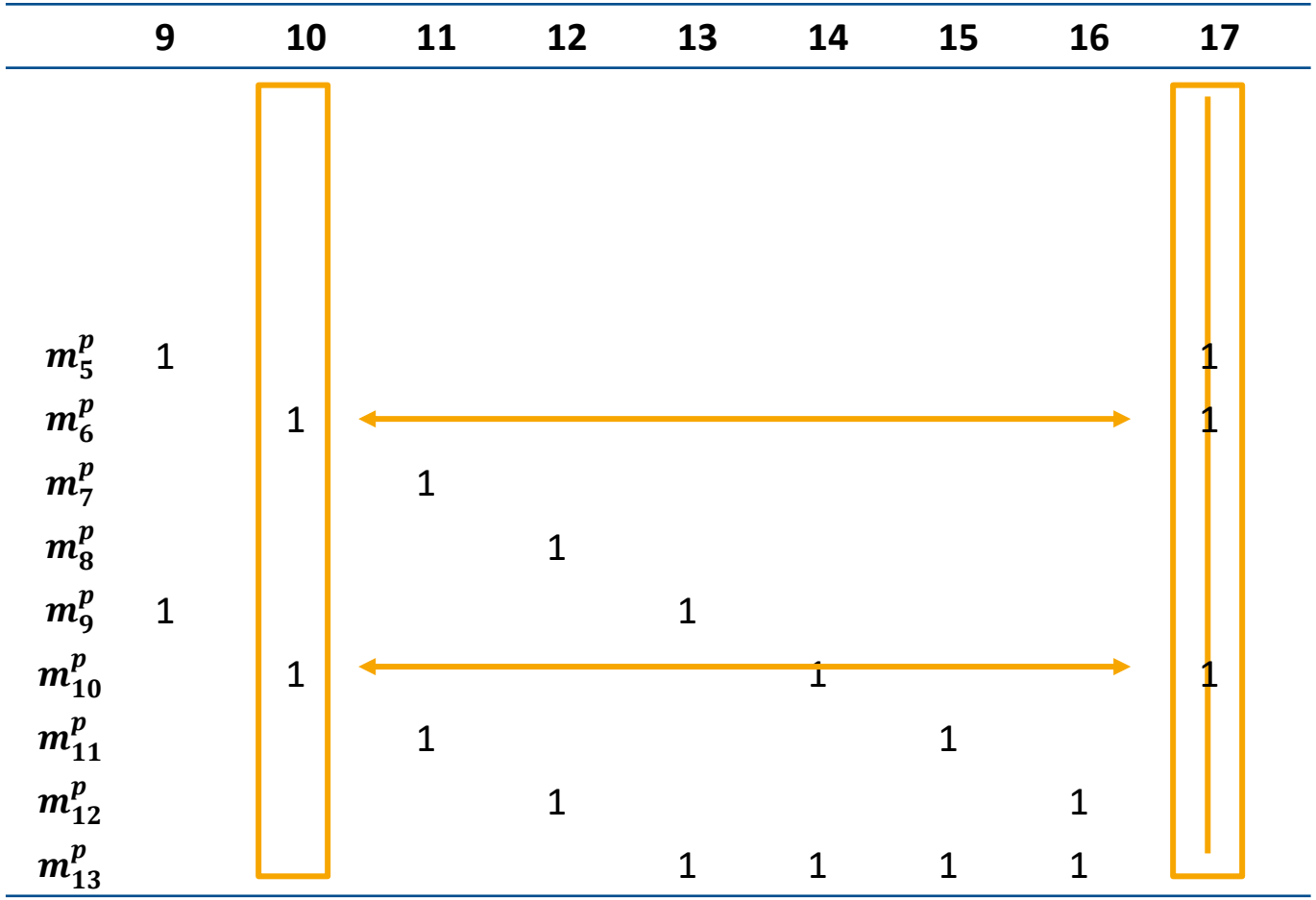
Zweite Reduktionsregel - Beispiel

- Spalte 17 dominiert Spalte 10



Zweite Reduktionsregel - Beispiel

- Spalte 17 dominiert Spalte 10
- Spalte 17 kann gelöscht werden



Zweite Reduktionsregel - Beispiel

- Spalte 17 dominiert Spalte 10
- Spalte 17 kann gelöscht werden

	9	10	11	12	13	14	15	16
m_5^p	1							
m_6^p		1						
m_7^p			1					
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel

Definition (dominante Spalten/Minterme):
Sei $A \in \mathbb{B}^{n \times m}$ eine Boolesche Matrix. Die Zeile i der Matrix A dominiert die Zeile l der Matrix A, wenn $a_{ij} \geq a_{lj}, \forall\, j \in \{1, \dots, m\}$.

Nutzen für unser Problem:

Dominiert ein Primimplikant m_i^p von f einen anderen Primimplikanten m_j^p von f, so braucht m_j^p nicht mehr weiter betrachtet zu werden, wenn $cost_1\left(m_j^p\right) \geq cost_1(m_i^p)$. Der Primimplikant m_i^p überdeckt jeden noch nicht überdeckten Minterm von f, der von m_j^p überdeckt wird, obwohl er nicht teurer ist.

Dritte Reduktionsregel:
Entferne aus der Primimplikantentafel $PIT(f)$ alle Primimplikanten, die von einem anderen, nicht teureren Primimplikanten dominiert werden.

Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

	9	10	11	12	13	14	15	16
m_5^p	1							
m_6^p		1						
m_7^p			1					
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5

	9	10	11	12	13	14	15	16
m_5^p	1							
m_6^p		1						
m_7^p			1					
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5
- Zeile 5 kann gelöscht werden

	9	10	11	12	13	14	15	16
m_6^p		1						
m_7^p			1					
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5
- Zeile 5 kann gelöscht werden

- Zeile 10 dominiert Zeile 6

	9	10	11	12	13	14	15	16
m_6^p		1						
m_7^p			1					
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5
- Zeile 5 kann gelöscht werden

- Zeile 10 dominiert Zeile 6
- Zeile 6 kann gelöscht werden

	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
m_7^p			1					
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5
- Zeile 5 kann gelöscht werden

- Zeile 10 dominiert Zeile 6
- Zeile 6 kann gelöscht werden

- Zeile 11 dominiert Zeile 7

	9	10	11	12	13	14	15	16
m_7^p			1					
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5
- Zeile 5 kann gelöscht werden

- Zeile 10 dominiert Zeile 6
- Zeile 6 kann gelöscht werden

- Zeile 11 dominiert Zeile 7
- Zeile 7 kann gelöscht werden

	9	10	11	12	13	14	15	16
m_8^p				1				
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

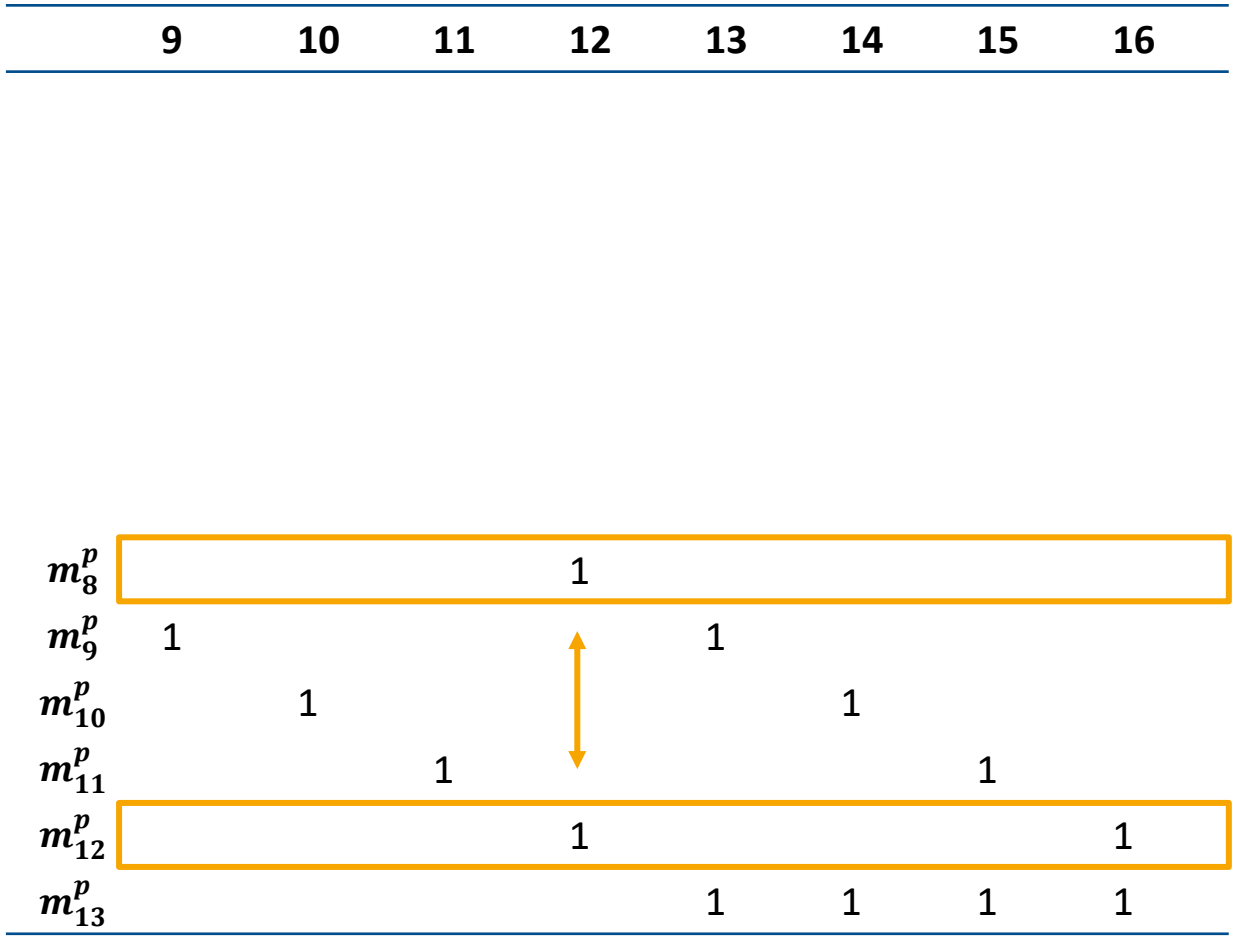
Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5
- Zeile 5 kann gelöscht werden

- Zeile 10 dominiert Zeile 6
- Zeile 6 kann gelöscht werden

- Zeile 11 dominiert Zeile 7
- Zeile 7 kann gelöscht werden

- Zeile 12 dominiert Zeile 8



Dritte Reduktionsregel - Beispiel

Annahme: Zeile 5 – 12 haben gleiche Kosten

- Zeile 9 dominiert Zeile 5
- Zeile 5 kann gelöscht werden

- Zeile 10 dominiert Zeile 6
- Zeile 6 kann gelöscht werden

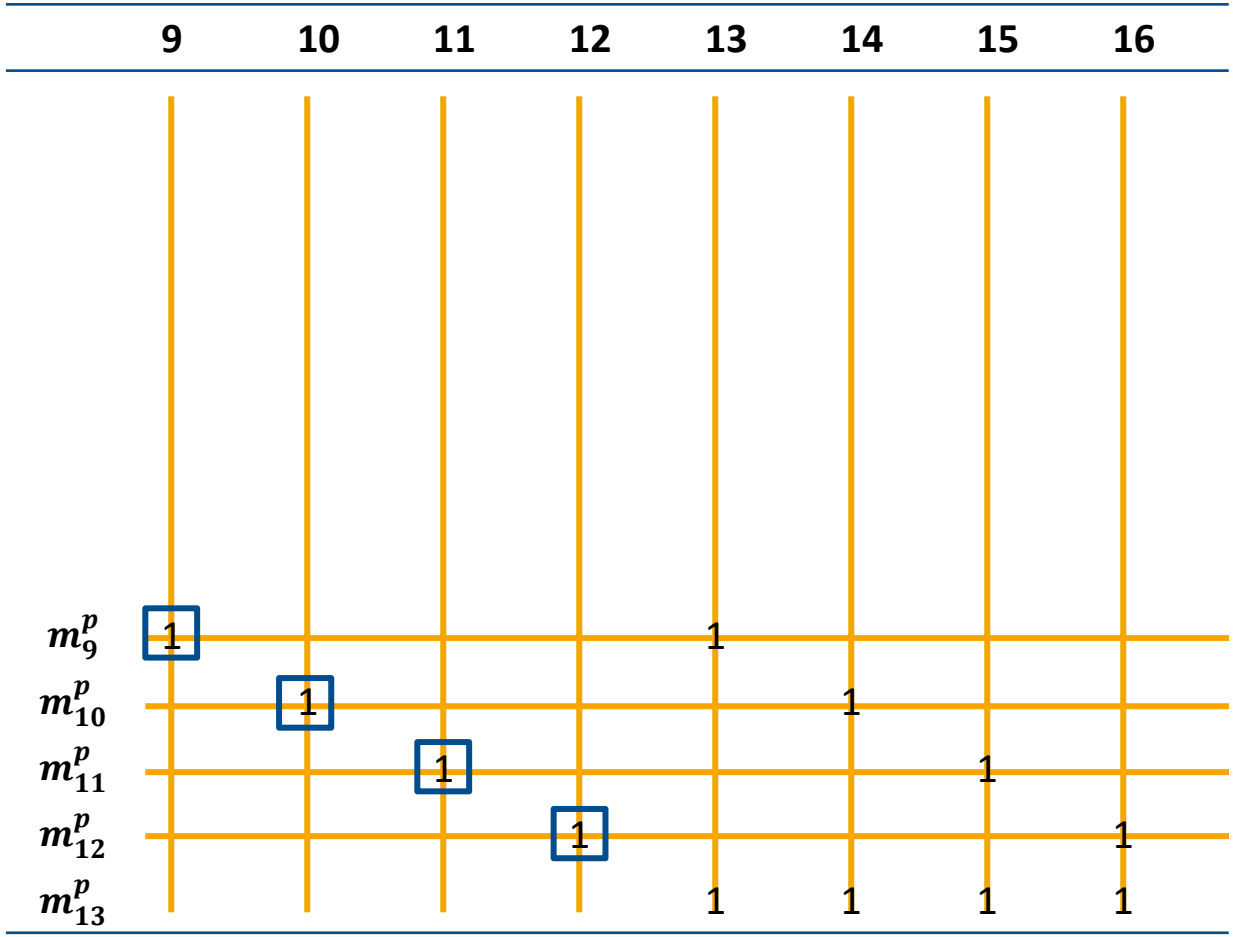
- Zeile 11 dominiert Zeile 7
- Zeile 7 kann gelöscht werden

- Zeile 12 dominiert Zeile 8
- Zeile 8 kann gelöscht werden

	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
m_9^p	1				1			
m_{10}^p		1				1		
m_{11}^p			1				1	
m_{12}^p				1				1
m_{13}^p					1	1	1	1

Dritte Reduktionsregel - Beispiel

- Wesentliche Primimplikanten (Regel 1) in
  - Zeile 9
  - Zeile 10
  - Zeile 11
  - Zeile 12
- Primimplikanten und Minterme können gelöscht werden
- Primimplikantentafel ist leer
- Die kostenminimale Teilmenge ist $M = \{m_1^p, m_2^p, m_3^p, m_4^p, m_9^p, m_{10}^p, m_{11}^p, m_{12}^p\}$
- Enthält nicht die Zeile mit den meisten Einsen



Zyklische Überdeckungsprobleme

Definition(reduzierte Primimplikantentafeln & zyklische Überdeckungsprobleme):
Eine Primimplikantentafel heißt reduziert, wenn keine der drei Reduktionsregeln anwendbar ist.
Ist eine reduzierte Tafel leer, entspricht die Menge M^* (siehe Reduktionsregel 1), der gesuchten kostenminimalen Menge.
Ist eine reduzierte Tafel nicht leer, spricht man von einem zyklischen Überdeckungsproblem.

Ansätze zum Lösen des zyklischen Überdeckungsproblems

- heuristische Verfahren (z.B. ESPRESSO)
- Verfahren von Petrick
- Branch-and-Bound Verfahren

	3	**5**	**7**	**9**	**11**	**13**
$\overline{x_1}x_2x_4$		1	1			
$\overline{x_1}x_3x_4$	1		1			
$\overline{x_2}x_3x_4$	1				1	
$x_1\overline{x_2}x_4$				1	1	
$x_1\overline{x_3}x_4$				1		1
$x_2\overline{x_3}x_4$		1				1

Verfahren von Petrick

	3	5	7	9	11	13
$a = \overline{x_1}x_2x_4$		1	1			
$b = \overline{x_1}x_3x_4$	1		1			
$c = \overline{x_2}x_3x_4$	1				1	
$d = x_1\overline{x_2}x_4$				1	1	
$e = x_1\overline{x_3}x_4$				1		1
$f = x_2\overline{x_3}x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

	3	**5**	**7**	**9**	**11**	**13**
$a = \overline{x_1}x_2x_4$		1	1			
$b = \overline{x_1}x_3x_4$	1		1			
$c = \overline{x_2}x_3x_4$	1				1	
$d = x_1\overline{x_2}x_4$				1	1	
$e = x_1\overline{x_3}x_4$				1		1
$f = x_2\overline{x_3}x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

$(b + c)$

	3	5	7	9	11	13
$a = \overline{x_1}x_2x_4$		1	1			
$b = \overline{x_1}x_3x_4$	1		1			
$c = \overline{x_2}x_3x_4$	1				1	
$d = x_1\overline{x_2}x_4$				1	1	
$e = x_1\overline{x_3}x_4$				1		1
$f = x_2\overline{x_3}x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

$(b + c) \cdot (a + f)$

	3	5	7	9	11	13
$a = \overline{x_1} x_2 x_4$		1	1			
$b = \overline{x_1} x_3 x_4$	1		1			
$c = \overline{x_2} x_3 x_4$	1				1	
$d = x_1 \overline{x_2} x_4$				1	1	
$e = x_1 \overline{x_3} x_4$				1		1
$f = x_2 \overline{x_3} x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

$(b + c) \cdot (a + f) \cdot (a + b)$

	3	5	7	9	11	13
$a = \overline{x_1} x_2 x_4$		1	1			
$b = \overline{x_1} x_3 x_4$	1		1			
$c = \overline{x_2} x_3 x_4$	1				1	
$d = x_1 \overline{x_2} x_4$				1	1	
$e = x_1 \overline{x_3} x_4$				1		1
$f = x_2 \overline{x_3} x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

$(b + c) \cdot (a + f) \cdot (a + b) \cdot (d + e)$

	3	5	7	9	11	13
$a = \overline{x_1} x_2 x_4$		1	1			
$b = \overline{x_1} x_3 x_4$	1		1			
$c = \overline{x_2} x_3 x_4$	1				1	
$d = x_1 \overline{x_2} x_4$				1	1	
$e = x_1 \overline{x_3} x_4$				1		1
$f = x_2 \overline{x_3} x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

$(b + c) \cdot (a + f) \cdot (a + b) \cdot (d + e) \cdot (c + d)$

	3	5	7	9	11	13
$a = \overline{x_1} x_2 x_4$		1	1			
$b = \overline{x_1} x_3 x_4$	1		1			
$c = \overline{x_2} x_3 x_4$	1				1	
$d = x_1 \overline{x_2} x_4$				1	1	
$e = x_1 \overline{x_3} x_4$				1		1
$f = x_2 \overline{x_3} x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

$$(b+c)\cdot(a+f)\cdot(a+b)\cdot(d+e)\cdot(c+d)\cdot(e+f)$$

	3	5	7	9	11	13
$a = \overline{x_1}x_2x_4$		1	1			
$b = \overline{x_1}x_3x_4$	1		1			
$c = \overline{x_2}x_3x_4$	1				1	
$d = x_1\overline{x_2}x_4$				1	1	
$e = x_1\overline{x_3}x_4$				1		1
$f = x_2\overline{x_3}x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme

$$(b + c) \cdot (a + f) \cdot (a + b) \cdot (d + e) \cdot (c + d) \cdot (e + f)$$

	3	**5**	**7**	**9**	**11**	**13**
$a = \overline{x_1} x_2 x_4$		1	1			
$b = \overline{x_1} x_3 x_4$	1		1			
$c = \overline{x_2} x_3 x_4$	1				1	
$d = x_1 \overline{x_2} x_4$				1	1	
$e = x_1 \overline{x_3} x_4$				1		1
$f = x_2 \overline{x_3} x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme
  - Schritt 2: Ausmultiplikation

$$(b+c)\cdot(a+f)\cdot(a+b)\cdot(d+e)\cdot(c+d)\cdot(e+f)$$
$$= badce + badcf + bade + badf + baec + baecf$$
$$+ baed + \ldots + bfd + \ldots + cae + \ldots + cfbed$$

	3	5	7	9	11	13
$a = \overline{x_1}x_2x_4$		1	1			
$b = \overline{x_1}x_3x_4$	1		1			
$c = \overline{x_2}x_3x_4$	1				1	
$d = x_1\overline{x_2}x_4$				1	1	
$e = x_1\overline{x_3}x_4$				1		1
$f = x_2\overline{x_3}x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme
  - Schritt 2: Ausmultiplikation
  - Schritt 3: Wähle kürzestes Monom

$$(b+c)\cdot(a+f)\cdot(a+b)\cdot(d+e)\cdot(c+d)\cdot(e+f)$$
$$= badce + badcf + bade + badf + baec + baecf$$
$$+ baed + \ldots + \boldsymbol{bfd} + \ldots + \boldsymbol{cae} + \ldots + cfbed$$

	3	**5**	**7**	**9**	**11**	**13**
$a = \overline{x_1}x_2x_4$		1	1			
$b = \overline{x_1}x_3x_4$	1		1			
$c = \overline{x_2}x_3x_4$	1				1	
$d = x_1\overline{x_2}x_4$				1	1	
$e = x_1\overline{x_3}x_4$				1		1
$f = x_2\overline{x_3}x_4$		1				1

Verfahren von Petrick

- Algorithmus:
  - Schritt 1: Bildung Produktsumme
  - Schritt 2: Ausmultiplikation
  - Schritt 3: Wähle kürzestes Monom

$$(b+c)\cdot(a+f)\cdot(a+b)\cdot(d+e)\cdot(c+d)\cdot(e+f)$$
$$= badce + badcf + bade + badf + baec + baecf$$
$$+ baed + \ldots + \boldsymbol{bfd} + \ldots + \boldsymbol{cae} + \ldots + cfbed$$

	3	5	7	9	11	13
$a = \overline{x_1}x_2x_4$		1	1			
$b = \overline{x_1}x_3x_4$	1		1			
$c = \overline{x_2}x_3x_4$	1				1	
$d = x_1\overline{x_2}x_4$				1	1	
$e = x_1\overline{x_3}x_4$				1		1
$f = x_2\overline{x_3}x_4$		1				1

Die Primimplikanten b, f, d oder die Primimplikanten c, a, e bilden eine kostenminimale Überdeckung für die Minterme 3,5,7,9,11,13