Kapitel 3 – Kombinatorische Logik

1. Kombinatorische Schaltkreise
2. Boolesche Algebren
3. Boolesche Ausdrücke, Normalformen, zweistufige Synthese
4. **Berechnung eines Minimalpolynoms**
5. Arithmetische Schaltungen
6. Anwendung: ALU von ReTI

Albert-Ludwigs-Universität Freiburg

Prof. Dr. Christoph Scholl

Institut für Informatik

WS 2015/16

Billigste Überdeckung der markierten Ecken

Wir suchen ein sogenanntes Minimalpolynom, das heißt ein Polynom mit minimalen Kosten.

Definition

Ein Minimalpolynom p einer booleschen Funktion f ist ein Polynom von f mit minimalen Kosten, das heißt mit der Eigenschaft $cost(p) \leq cost(p')$ für jedes andere Polynom p' von f.

$\psi(p) = f$

Quine's Primimplikantensatz

Satz

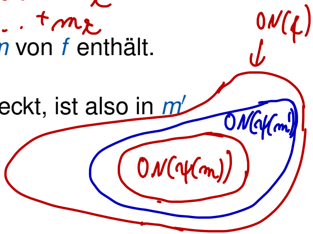
Jedes Minimalpolynom p einer booleschen Funktion f besteht ausschließlich aus Primimplikanten von f.

$p = m_1 + \dots + m_{i-1} + \underline{m} + m_{i+1} + \dots + m_r$

$p' = m_1 + \dots + m_{i-1} + \underline{m'} + m_{i+1} + \dots + m_r$

Beweis:

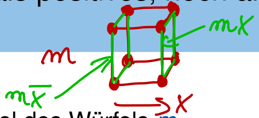
- Nehme an, dass p einen nicht primen Implikanten m von f enthält.
- m wird durch einen Primimplikanten m' von f überdeckt, ist also in m' enthalten.
- Es gilt demnach $cost(m') < cost(m)$.
- Ersetzt man in p den Implikanten m durch den Primimplikanten m', so erhält man ein Polynom p', das ein Polynom von f ist mit $cost(p') < cost(p)$.
- **Widerspruch** dazu, dass p ein Minimalpolynom ist.



Berechnung von Implikanten

Lemma 1

Ist m ein Implikant von f, so auch $m \cdot x$ und $m \cdot x'$ für jede Variable x, die in m weder als positives, noch als negatives Literal vorkommt.



Beweis:

- $m \cdot x$ und $m \cdot x'$ sind Teilwürfel des Würfels m.
- Sind also alle Ecken von m markiert, so auch alle Ecken von $m \cdot x$ und $m \cdot x'$.

Lemma 2

Sind $m \cdot x$ und $m \cdot x'$ Implikanten von f, so auch m.

Beweis: ...

$$\left.\begin{array}{l} mx \text{ Impl. von } f \Rightarrow ON(\psi(mx)) \subseteq ON(f) \\ m\bar{x} \text{ '' '' ''} \Rightarrow ON(\psi(m\bar{x})) \subseteq ON(f) \end{array}\right\} \Rightarrow$$

$$ON(\psi(mx)) \cup ON(\psi(m\bar{x})) \subseteq ON(f)$$

$$\Rightarrow ON(\psi(mx) \vee \psi(m\bar{x})) \subseteq ON(f)$$

$$\Rightarrow ON(\psi(m)\cdot\psi(x) \vee \psi(m)\cdot\psi(\bar{x})) \subseteq ON(f)$$

$$\Rightarrow ON(\psi(m)\cdot(\underbrace{\psi(x) \vee \overline{\psi(x)}}_{1})) \subseteq ON(f)$$

$$\Rightarrow ON(\psi(m)) \subseteq ON(f)$$

$\Rightarrow$ m ist Implikant von f

Charakterisierung von Implikanten

Satz

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.

- Äquivalente Schreibweise:

  $m \in Implikant(f)$
  $\Leftrightarrow (m \in Minterm(f)) \vee (m \cdot x, m \cdot x' \in Implikant(f))$

- Beweis folgt unmittelbar aus Lemma 1 und Lemma 2.

„$\Rightarrow$“ „$\Leftarrow$“

Berechnung eines Minimalpolynoms

- Verfahren von Quine-McCluskey zur Berechnung aller Primimplikanten.

  Idee: Berechne sogar alle Implikanten. Dann wird klar, welche davon Primimpl. sind.
- Verfahren zur Lösung des „Überdeckungsproblems".
  - Treffe unter den Primimplikanten eine geeignete Auswahl, so dass die Disjunktion der ausgewählten Primimplikanten ein Polynom für f ist und minimale Kosten hat.

Verfahren von Quine: Der Algorithmus

Prime implicants function **Quine** ($f : \mathbb{B}^n \to \mathbb{B}$)

begin
$L_0 := Minterm(f)$; Menge aller Minterme von f = Menge aller Implikanten der Länge n
$i := 0$;
$Prim(f) := \emptyset$
while ($L_i \neq \emptyset$) **and** ($i < n$)
$//L_i$ enthält alle Implikanten von f der Länge $n - i$.
loop $L_{i+1} := \{m \mid m \cdot x$ und $m \cdot x'$ sind in L_i für ein $x\}$;
$Prim(f) := Prim(f) \cup$
$\{m' \mid m' \in L_i$ und m' wird von keinem $q \in L_{i+1}$ überdeckt$\}$;
$i := i + 1$
end loop;
return $Prim(f) \cup L_i$;
end;

bzw. es gibt kein x, so dass
$m' = qx \in L_i$ und $q\bar{x} \in L_i$, so dass
$q \in L_{i+1}$ oder
$m' = q\bar{x} \in L_i$ und $qx \in L_i$, so dass
$q \in L_{i+1}$.

8

Verbesserung durch McCluskey

$\underline{m x}, \underline{m \bar{x}}$

- Vergleiche nur Monome untereinander
  - die die gleichen Variablen enthalten und
  - bei denen sich die Anzahl der positiven Literale nur um 1 unterscheidet.

$L_1 = \{x_1\bar{x}_2x_3, \bar{x}_1\bar{x}_2x_3, x_3x_4\bar{x}_5\}$

$L_1^{\{x_1,x_2,x_3\}} = \{x_1\bar{x}_2x_3, \bar{x}_1\bar{x}_2x_3\}$ $L_1^{\{x_3,x_4,x_5\}} = \{x_3x_4\bar{x}_5\}$

- Dies wird erreicht durch:
  - Partitionierung von L_i in Klassen L_i^M, mit $M \subseteq \{x_1, \ldots, x_n\}$ und $|M| = n - i$.

$L_1^{\{x_1,x_3,x_4\}} = \emptyset$

  - L_i^M enthält die Implikanten aus L_i, deren Literale alle aus M sind.
  - Anordnung der Monome in L_i^M gemäß der Anzahl der positiven Literale.

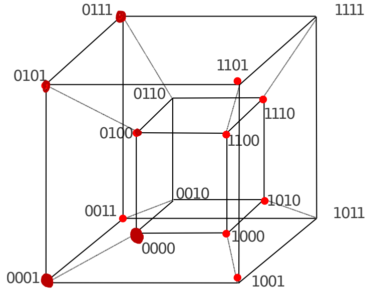
9

Beispiel Quine-McCluskey

$f: \{0,1\}^4 \to \{0,1\}$

$L_0 = \{\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \mid \bar{x}_1\bar{x}_2\bar{x}_3 x_4 \mid \dots\}$

$\left.\begin{array}{l} m\,x \\ m\,\bar{x} \end{array}\right\} \Rightarrow m$



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

0 pos. Lit.
- 0 0 0 0 — steht für $x_1'x_2'x_3'x_4'$

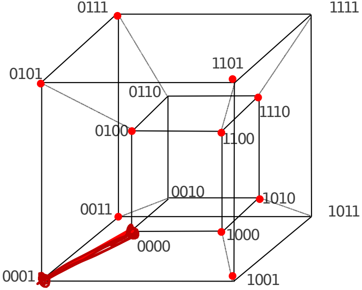
1 pos. Lit.
- 0 0 0 1
- 0 1 0 0
- 1 0 0 0

2 pos. Lit.
- 0 0 1 1 — $\bar{x}_1\bar{x}_2 x_3 x_4$
- 0 1 0 1
- 1 0 0 1 — steht für $x_1x_2'x_3'x_4$
- 1 0 1 0
- 1 1 0 0

3 pos. Lit.
- 0 1 1 1
- 1 1 0 1
- 1 1 1 0

Vergleiche im Folgenden nur Monome aus benachbarten Blöcken!

Beispiel Quine-McCluskey: Bestimmung von L_1 (1/4)



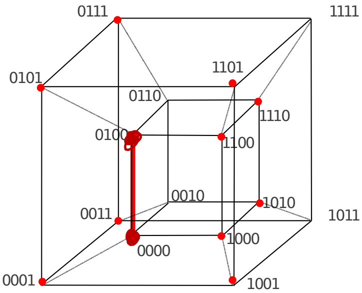
$L_0^{\{x_1,x_2,x_3,x_4\}}$:

$L_0^{\{x_1,x_2,x_3,x_4\}}$		$L_1^{\{x_1,x_2,x_3\}}$
0 0 0 0	$\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}$	0 0 0 -
0 0 0 1	$\underbrace{\overline{x_1}\,\overline{x_2}\,\overline{x_3}}_{m}\,x_4$	
0 1 0 0		
1 0 0 0		
0 0 1 1	$\overline{x_1}\,\overline{x_2}\,\overline{x_3}$	
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		

statt für $\overline{x_1}\,\overline{x_2}\,\overline{x_3}$

Beispiel Quine-McCluskey: Bestimmung von L_1 (2/4)

„Wähle $m = \overline{x_1}\,\overline{x_3}\,\overline{x_4}$, $x = x_2$"



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

0 0 0 0 ← $\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}$
0 0 0 1
0 1 0 0 ← $\overline{x_1}\,x_2\,\overline{x_3}\,\overline{x_4}$
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

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

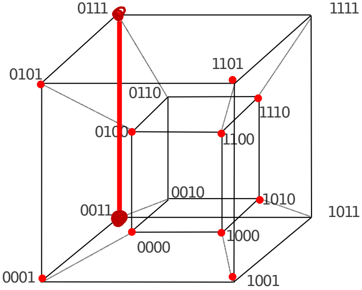
0 0 0 -

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

0 - 0 0 ← $\overline{x_1}\,\overline{x_3}\,\overline{x_4}$

$\overline{x_1}\,\overline{x_3}\,\overline{x_4}$

Beispiel Quine-McCluskey: Bestimmung von L_1 (3/4)



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

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 -

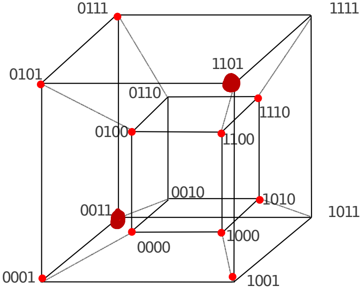
$L_1^{\{x_1,x_3,x_4\}}$:

0 - 0 0

0 - 1 1 ← $\hat{=}\ \overline{x_1}\, x_3 x_4$

-13-

Beispiel Quine-McCluskey: Bestimmung von L_1 (4/4)



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

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

$\bar{x}_1 \bar{x}_2 x_3 x_4$

$x_1 x_2 \bar{x}_3 x_4$

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

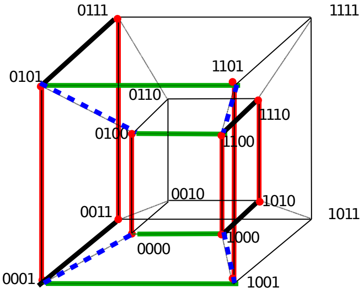
0 0 0 -

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

0 - 0 0
0 - 1 1

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

Beispiel Quine-McCluskey: Alle bestimmten Mengen L_1



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

0 0 - 1
1 0 - 0

---

0 1 - 1
1 1 - 0

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

0 0 0 -

---

0 1 0 -
1 0 0 -

---

1 1 0 -

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

- 0 0 0

- 0 0 1
- 1 0 0

- 1 0 1

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

0 - 0 0

---

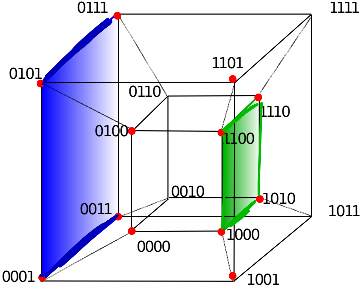
0 - 0 1
1 - 0 0

---

0 - 1 1
1 - 0 1
1 - 1 0

Alle Minterme von f sind Eckpunkte von Kanten, die Implikanten sind: $Prim(f) = \emptyset$

Beispiel Quine-McCluskey: Bestimmung von L_2 (1/2)



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

0 0 - 1	$\overline{x_1}\,\overline{x_2}\,x_4$
1 0 - 0	
0 1 - 1	$\overline{x_1}\,x_2\,x_4$
1 1 - 0	

$x_1 x_2 \overline{x_4}$ $\qquad$ $\overline{x_1}\,x_4$

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

0 0 0 -
0 1 0 -
1 0 0 -
1 1 0 -

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

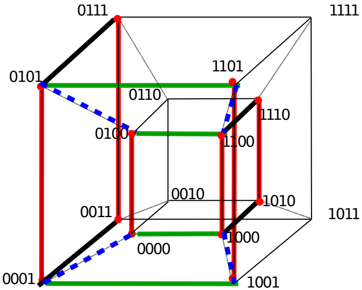
- 0 0 0
- 0 0 1
- 1 0 0
- 1 0 1

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

0 - 0 0
0 - 0 1
1 - 0 0
0 - 1 1
1 - 0 1
1 - 1 0

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

Beispiel Quine-McCluskey: Bestimmung von L_2 (2/2)



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

0 0 - 1
1 0 - 0
0 1 - 1
1 1 - 0

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

0 0 0 -
0 1 0 -
1 0 0 -
1 1 0 -

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

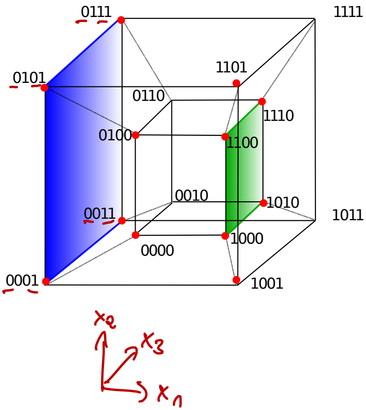
- 0 0 0
- 0 0 1
- 1 0 0
- 1 0 1

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

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: $Prim(f) = \emptyset$

Beispiel Quine-McCluskey: Bestimmung von L_3 (1/2)



$L_2^{\{x_1,x_2\}}$:

$L_2^{\{x_1,x_3\}}$:

$\bar{x}_1\bar{x}_3$ 0 - 0 -

$x_1\bar{x}_3$ 1 - 0 - $\to \bar{x}_3$

$L_2^{\{x_1,x_4\}}$:

0 - - 1 $\leftarrow \bar{x}_1 x_4$

1 - - 0 $\leftarrow x_1 \bar{x}_4$

$L_2^{\{x_2,x_3\}}$:

- 0 0 -

- 1 0 - $\to \bar{x}_3$

$L_2^{\{x_2,x_4\}}$:

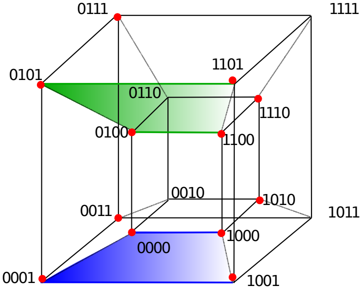
$L_2^{\{x_3,x_4\}}$:

- - 0 0

- - 0 1 $\to \bar{x}_3$

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

Beispiel Quine-McCluskey: Bestimmung von L_3 (2/2)



$L_2^{\{x_1,x_2\}}$:

$L_2^{\{x_1,x_4\}}$:

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 -

$L_2^{\{x_3,x_4\}}$:

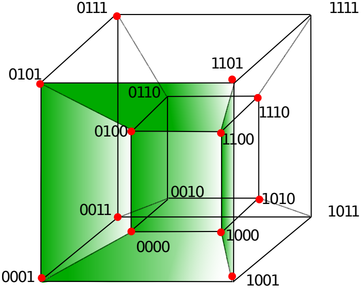
- - 0 0

- - 0 1

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

19

Beispiel Quine-McCluskey: Ende



$L_3^{\{x_1\}}$:

$L_3^{\{x_2\}}$:

$L_3^{\{x_3\}}$:

$L_3^{\{x_4\}}$:

- - 0 - $\overline{x_3}$

$L_4 = \emptyset$

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

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

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

20

Korrektheit von Quine-McCluskey (1/2)

Prime implicants function **Quine** ($f : \mathbb{B}^n \to \mathbb{B}$)

begin
  $L_0 := Minterm(f)$;
  $i := 0$;
  $Prim(f) := \emptyset$
  while ($L_i \neq \emptyset$) **and** ($i < n$)
  $//L_i$ enthält alle Implikanten von f der Länge $n - i$.
  loop $L_{i+1} := \{m \mid m \cdot x$ und $m \cdot x'$ sind in L_i für ein $x\}$;
    $Prim(f) := Prim(f) \cup$
    $\{m' \mid m' \in L_i$ und m' wird von keinem $q \in L_{i+1}$ überdeckt$\}$;
    $i := i + 1$
  end loop;
  return $Prim(f) \cup L_i$;
end;

Korrektheit von Quine-McCluskey (2/2)

Satz

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 mindestens $n - i$ Literalen.

Beweis:
Induktion über i:

- Abbruchbedingung $(L_i = \emptyset)$ oder $(i = n)$:
- $L_i = \emptyset$ bedeutet, dass keine Implikanten bei der "Partnersuche" entstanden sind, d.h. L_{i-1} ist vollständig 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 $Prim(f) = \{1\}$.

22

Kosten des Verfahrens

Frage: Wie viele Schritte braucht der Algorithmus?
Einfacher: Wieviele Implikanten erzeugt man im schlimmsten Fall?

Lemma

Es gibt 3^n verschiedene Monome in n Variablen.

$x_1 x_2 x_3 \ldots x_n$

$3 \cdot 3 \cdot 3$

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 x'.

Jedes Monom ist durch diese Beschreibung auch eindeutig bestimmt.

Komplexität des Verfahrens von Quine-McCluskey

Satz

Die Laufzeit des Verfahrens liegt in $O(n^2 \cdot 3^n)$ beziehungsweise in $O(\log^2(N) \cdot N^{\log(3)})$, wobei $N = 2^n$ die Größe der Funktionstabelle ist.

$\to mx$

$m\bar{x}$

Beweisidee:
Jedes der 3^n Monome wird im Verlauf des Verfahrens mit höchstens n anderen Monomen verglichen.

- Gegeben sei ein Monom mx. Die Erzeugung von mx' und die Suche nach mx' in L_i ist bei Verwendung geeigneter Datenstrukturen in $O(n)$ durchführbar.

$O(n^2 \cdot 3^n) = O(\log^2(N) \cdot N^{\log(3)})$ durch Nachrechnen: $N = 2^n \Rightarrow \log_2 N = n$

$$3^n = \underbrace{(2^{\log_2(3)})}_{3}{}^n = (2^n)^{\log_2(3)} = N^{\log(3)} \approx N^{1.58}$$

$(x^a)^b = x^{ab} = (x^b)^a$

Erinnerung:

$f(x) \in O(g(x)) \Longleftrightarrow c \in \mathbb{R}_0^+, x_0 \in \mathbb{R}_0^+$ mit $f(x) \leq c \cdot g(x) \quad \forall x > x_0$.

Anwendung:

Laufzeitfunktion $f(N)$ nach oben abschätzen

z.B.: $f(x) = \underline{c_1} \cdot x^{a_1} + c_2 \cdot x^{a_2} + \ldots + c_n \cdot x^{a_n} \in O(x^{\underline{a_0}})$

mit $a_0 \geq a_1 > a_2 > a_3 \ldots > a_n, \; c_1 > 0$.

$\Longrightarrow a_0 = a_1$

Das Matrix-Überdeckungsproblem

- Wir haben nun durch das Verfahren von Quine-McCluskey alle Primimplikanten von f bestimmt.
- Die Disjunktion aller Primimplikanten ist ein Polynom, das f implementiert. Es ist aber im Allgemeinen kein Minimalpolynom von f.
- Für das Minimalpolynom benötigen wir eine kostenminimale Teilmenge M von $Prim(f)$, so dass die Monome von M f überdecken.
- Diese Art von Problemen wird Matrix-Überdeckungsproblem genannt.

Bsp.: $\mathrm{Prim}(f) = \{ x_1\overline{x_2}, x_2x_3, x_1x_3 \}$.

Werden alle Primimpl. von $\mathrm{Prim}(f)$ gebraucht im Minimalpolynom?

$$\underline{x_1\overline{x_2}} + \underline{x_2x_3} + x_1x_3 \equiv x_1\overline{x_2} + x_2x_3$$

Primimplikantentafel

P.i.

m

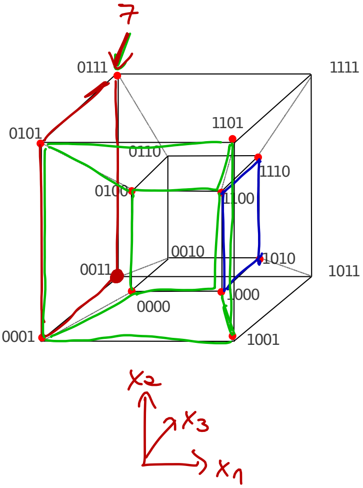
	$min(\alpha_1)$	$min(\alpha_2)$	...	$min(\alpha_l)$
m_1				
m_2	0	1	0 ...	1
$\vdots$				
m_k				

- 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. bzw. den Eins-stellen von f. $ON(f) = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}$
  - Sei $min(\alpha)$ ein beliebiger Minterm von f.
    Dann gilt für Primimplikant m: $PIT(f)[m, min(\alpha)] = 1 \Leftrightarrow m(\alpha) = 1$.
- 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 $Prim(f)$, so dass jede Spalte von $PIT(f)$ überdeckt ist,

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

Primimplikantentafel: Beispiel (1/2)



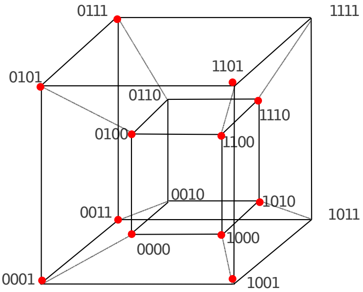
$Prim(f) = \{x_1' x_4, x_1 x_4', x_3'\}$

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	

Primimplikantentafel: Beispiel (2/2)

Gesucht:
Eine kostenminimale Teilmenge M von $Prim(f)$, so dass jede Spalte von $PIT(f)$ überdeckt ist,
d.h. $\forall \alpha \in ON(f) \quad \exists m \in M$ mit $PIT(f)[m, min(\alpha)] = 1$.



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

0011

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	

Erste Reduktionsregel - Wesentlicher Implikant

Definition

Ein Primimplikant m von f heißt wesentlich, wenn es einen Minterm $min(\alpha)$ von f gibt, der nur von diesem Primimplikanten überdeckt wird, also:

- $PIT(f)[m, min(\alpha)] = 1$
- $PIT(f)[m', min(\alpha)] = 0$

für jeden anderen Primimplikanten m' von f.

Lemma

Jedes Minimalpolynom von f enthält alle wesentlichen Primimplikanten von f.

1. Reduktionsregel: Entferne aus der Primimplikantentafel $PIT(f)$ alle wesentlichen Primimplikanten und alle Minterme, die von diesen überdeckt werden.

31

Erste Reduktionsregel: Beispiel (1/2)

Elemente von ON(f) (Minterme)

P.i.

	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: Beispiel (2/2)

wesentlich

	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	

Nach Anwendung der 1. 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 - Spaltendominanz

Definition

Sei A eine boolesche Matrix. Spalte j von A **dominiert** Spalte i von A, wenn für jede Zeile k gilt: $A[k,i] \le A[k,j]$.

$$\begin{array}{cc} i & j \\ \left[\begin{array}{c} 0 \\ 1 \\ 1 \\ 0 \end{array}\right. & \left.\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \end{array}\right] \\ \text{dominiert} & \text{dominierend} \end{array}$$

dominierende Spalte kann man weg= lassen!

- Nutzen für unser Problem: Dominiert ein Minterm w' von f einen anderen Minterm w von f, so braucht man w' nicht weiter zu betrachten, da w auf jeden Fall überdeckt werden muss und hierdurch automatisch auch Minterm w' überdeckt wird.
- Jeder in $PIT(f)$ vorhandene Primimplikant p, der w überdeckt, überdeckt auch w'.

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

Zweite Reduktionsregel: Beispiel

	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 - Zeilendominanz

Definition

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

(m) dominierend → i   1 1 1 1 0 0

(m') dominiert → j   0 1 0 1 0 0

- Nutzen für unser Problem: Dominiert ein Primimplikant m einen Primimplikanten m', so braucht man m' nicht weiter zu betrachten, wenn $cost(m') \geq cost(m)$ gilt.

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

3. Reduktionsregel: Entferne aus der Primimplikantentafel $PIT(f)$ alle Primimplikanten, die durch einen anderen, nicht teureren Primimplikanten dominiert werden.

Dritte Reduktionsregel: Beispiel

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: Beispiel

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

werden dominiert

	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

Nach Anwendung der 3. 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:
    $1 + 2 + 3 + 4 + 9 + 10 + 11 + 12$

Ein weiteres Beispiel

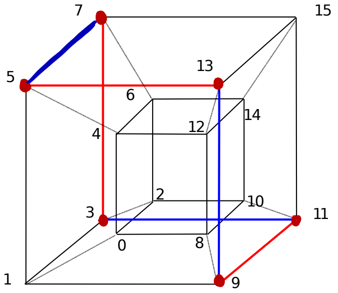
$Prim(f) = \{\{7,5\},\{5,13\},\{13,9\},\{9,11\},\{11,3\},\{3,7\}\}$

Primimplikantentafel *PIT*(f):

	3	5	7	9	11	13
$\{7,5\}$	0	1	1	0	0	0
$\{5,13\}$		1				1
$\{13,9\}$				1		1
$\{9,11\}$				1	1	
$\{11,3\}$	1				1	
$\{3,7\}$	1		1			

Wie sieht die kostenminimale Lösung aus?

Ein weiteres Beispiel



$Prim(f) = \{\{7,5\},\{5,13\},\{13,9\},\{9,11\},\{11,3\},\{3,7\}\}$

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!

Außerdem: Reduktionsregeln 2 und 3 sind auch nicht anwendbar.

$$Prim(f) = \{\boxed{\{7,5\}}, \boxed{\{5,13\}}, \boxed{\{13,9\}}, \boxed{\{9,11\}}, \boxed{\{11,3\}}, \boxed{\{3,7\}}\}$$

Wie sieht die kostenminimale Lösung aus?

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.
- In der Praxis werden solche Probleme heuristisch gelöst. Es gibt auch exakte Methoden (Petrick, Branch-and-Bound).

suche eine möglichst gute Lösung

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		7			

UNI FREIBURG

43

Petrick's Methode

Boolesche Variable a mit a = 1 ⟺ d.h. {7,5} in der Lösung verwendet wird.

Verfahren:

- Übersetze die PIT in ein (OR, AND)-Polynom, das alle Möglichkeiten der Überdeckung enthält.
- Multipliziere das (OR, AND)-Polynom aus, so dass ein (AND-OR)-Polynom entsteht.
- Die gesuchte minimale Überdeckung ist gegeben durch das Monom, das einer PI-Auswahl mit minimalen Kosten entspricht.

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

	3	5	7	9	11	13
a : {7,5}		1	1			
b : {5,13}		1				1
c : {13,9}				1		1
d : {9,11}				1	1	
e : {11,3}	1				1	
f : {3,7}	1		1			

Konjunktion aller Spaltenbedingungen

wird übersetzt in

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

$$= (ea+eb+fa+fb)\cdot(ac+ad+fc+fd)$$
$$\cdot(db+dc+eb+ec)$$
$$\vdots$$
$$= ace + acde + abcde + abcd + \cdots + bdf$$

Jede 1-Stelle dieses OR-AND-Polynoms entspricht einer gültigen Überdeckung der 1-Stellen der Funktion f durch Primimplikanten

Bei gleichen Kosten für alle PIs sind *ace* und *bdf* minimal.

1. Wende alle möglichen Reduktionsregeln an.
2. Ist die Matrix A leer, ist man fertig.
3. Sonst wähle die Zeile i, die die meisten Spalten überdeckt. Lösche diese Zeile und alle von ihr überdeckten Spalten und gehe zu 1.

- Dieser Algorithmus liefert nicht immer die optimale Lösung!
  - Hinweis: Bei der Ausgangs-Matrix aus unserem Beispiel überdeckt Zeile 13 die meisten Spalten.
    Diese ist nicht Teil der gefundenen Lösung!

Zusammenfassung Schaltkreise

- Schaltkreise stellen boolesche Funktionen dar.
- Boolesche Polynome kann man als eingeschränkte Schaltkreise betrachten. Dafür gibt es exakte Minimierungsverfahren.
- Optimale boolesche Polynome können sehr viel größer sein, als entsprechende Schaltkreise.
  - exponentielle Unterschiede möglich
  - Rechtfertigung für Einsatz von Schaltkreisen statt PLAs
- Es gibt auch Algorithmen zur Berechnung optimaler (mehrstufiger) Schaltkreise.
  - anspruchsvoller als Optimierung von booleschen Polynomen
  - meist heuristisch (Näherungsverfahren)
  - nicht Gegenstand dieser Vorlesung
- Hier: Schaltkreise für spezielle Funktionen, insbesondere Arithmetik.

46

Beispiel:

$\mathrm{xor}_n \in \mathbb{B}_n$ mit

$$\mathrm{xor}_n(x_1, \ldots, x_n) = \left(\sum_{i=1}^{n} x_i\right) \bmod 2 = \begin{cases} 1, & \text{falls } \sum_{i=1}^{n} x_i \text{ ungerade} \\ 0, & \text{falls } \sum_{i=1}^{n} x_i \text{ gerade} \end{cases}$$

Frage: Wie sieht Minimalpolynom für xor_n aus?

- Wie sieht die ON-Menge von xor_n aus / wieviele Elemente?
  Die Hälfte aller Elemente von $\{0,1\}^n = \mathbb{B}^n$ gehört zur ON-Menge.
  $\Rightarrow$ 2^{n-1} Elemente in der ON-Menge.

- Berechne die P.i. durch Algorithmus von Quine/McCluskey.
  Anfang: Beginne mit allen Mintermen (2^{n-1} Stück).
  $mx, m\overline{x}$
  $\Rightarrow$ Beim Algorithmus von Q.-McC. lassen sich gar keine Minterme „kombinieren" $\Rightarrow$ Alle Minterme sind P.i.
  $\Rightarrow$ Alle P.i. sind wesentlich
  $\Rightarrow$ Minimalpolynom besteht aus 2^{n-1} Monomen der Länge n.

$$\mathrm{xor}_n(x_1,\dots,x_n) = g\big(\underbrace{\mathrm{xor}_s(x_1,\dots,x_s)}_{e_1},\ \underbrace{\mathrm{xor}_{n-s}(\underline{x_{s+1},\dots,x_n})}_{e_2}\big)$$

e_1	e_2	g
0	0	0
0	1	1
1	0	1
1	1	0

$$= \mathrm{xor}_2\big(\mathrm{xor}_s(x_1,\dots,x_s),\ \mathrm{xor}_{n-s}(x_{s+1},\dots,x_n)\big)$$

$(g = \mathrm{xor}_2)$

<u>1. Fall</u>: $e_1 = 0,\ e_2 = 0$

$\sum_{i=1}^{s} x_i$ gerade $\quad \sum_{i=s+1}^{n} x_i$ gerade

$\sum_{i=1}^{n} x_i$ gerade

<u>2. Fall</u>: $e_1 = 0,\ e_2 = 1$

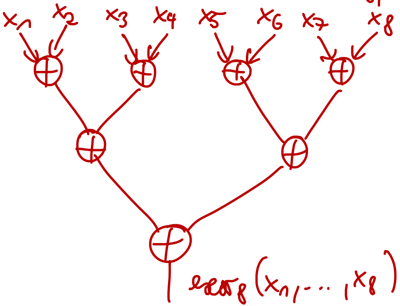
$\sum_{i=1}^{s} x_i$ gerade, $\quad \sum_{i=s+1}^{n} x_i$ ungerade

$\Longrightarrow \sum_{i=1}^{n} x_i = \sum_{i=1}^{s} x_i + \sum_{i=s+1}^{n} x_i$ ungerade

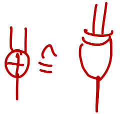
$\Longrightarrow$ Schaltkreis ~~für~~ für xor_n mit $n = 8$:

$$xor_8(x_1, \dots, x_8) = xor_2(xor_4(x_1, \dots, x_4), xor_4(x_5, \dots, x_8))$$

$$= xor_2(xor_2(xor_2(x_1, x_2), xor_2(x_3, x_4)), xor_2(xor_2(x_5, x_6), xor_2(x_7, x_8)))$$



$$\boxed{\begin{aligned} xor_3(x_1, x_2, x_3) &= \\ & xor_2(xor_2(x_1, x_2), xor_1(x_3)) \\ &= xor_2(xor_2(x_1, x_2), x_3) \end{aligned}}$$



Im allgemeinen Fall: $\underline{n-1}$ xor_2-Gatter, längster Pfad: $\lceil \log n \rceil$