technische universität
dortmund

fakultät für informatik

Rechnerstrukturen, Teil 1

Vorlesung 4 SWS WS 19/20

Prof. Dr. Jian-Jia Chen
Fakultät für Informatik – Technische Universität Dortmund
jian-jia.chen@cs.uni-dortmund.de
http://ls12-www.cs.tu-dortmund.de

Übersicht

4. **Boolesche Funktionen und Schaltnetze**
    1. **Einleitung**
    2. Boolesche Algebra
    3. Repräsentationen boolescher Funktionen
    4. Normalformen boolescher Funktionen
    5. Repräsentation boolescher Funktionen mit OBDDs
    6. Schaltnetze

4.1 Einleitung

Boolesche Funktionen

Vielleicht schon bekannt: **Aussagenlogik (engl. Propositional calculus)**

- **Satz** ist Aussage mit eindeutigem Wahrheitswert

- Wahrheitswerte
  - **wahr**
  - **falsch**

- **neue Aussagen** entstehen durch **Verknüpfung** von Aussagen

- Verknüpfungen
  - **Negation** ($\neg$, "nicht")
  - **Konjunktion** ($\wedge$, "und")
  - **Disjunktion** ($\vee$, "oder")

4.1 Einleitung

Definition der Verknüpfungen

Seien A, B zwei Aussagen.

Definition Negation

A	$\neg A$
falsch	wahr
wahr	falsch

Definition Konjunktion

A	B	$A \wedge B$
falsch	falsch	falsch
falsch	wahr	falsch
wahr	falsch	falsch
wahr	wahr	wahr

Definition Disjunktion

A	B	$A \vee B$
falsch	falsch	falsch
falsch	wahr	wahr
wahr	falsch	wahr
wahr	wahr	wahr

4. Boolesche Funktionen und Schaltnetze

1. Einleitung ✓
2. **Boolesche Algebra**
3. Repräsentationen boolescher Funktionen
4. Normalformen boolescher Funktionen
5. Repräsentation boolescher Funktionen mit OBDDs
6. Schaltnetze

4.2 Boolesche Algebra

Eine **boolesche Algebra** ist eine spezielle algebraische Struktur, die die

- Eigenschaften der **logischen Operatoren** **UND**, **ODER**, **NICHT** und die
- Eigenschaften der **mengentheoretischen Verknüpfungen** **Durchschnitt**, **Vereinigung**, **Komplement**

verallgemeinert.

Definition 2

Wir nennen $(B, \cup, \cap, \bar{\ })$ mit $B = \{0, 1\}$ und

- $x \cup y = max(x, y)$
- $x \cap y = min(x, y)$
- $\bar{x} = 1 - x$

für alle $x, y \in B$ eine **boolesche Algebra**.

4.2 Boolesche Algebra

Wir sehen hier bereits **Entsprechungen** zur **Aussagenlogik**:

- falsch $\Leftrightarrow$ 0
- wahr $\Leftrightarrow$ 1
- $\wedge \Leftrightarrow \cap$
- $\vee \Leftrightarrow \cup$
- $\neg \Leftrightarrow \overline{}$

4.2 Boolesche Algebra

In einer **boolesche Algebra** gelten folgende Rechengesetze:

Satz 3

In der booleschen Algebra $(B, \cup, \cap, \bar{\ })$ für alle $x, y, z \in B$:

Kommutativität:
$$x \cup y = y \cup x$$
$$x \cap y = y \cap x$$

Assoziativität:
$$(x \cup y) \cup z = x \cup (y \cup z)$$
$$(x \cap y) \cap z = x \cap (y \cap z)$$

Distributivität:
$$x \cap (y \cup z) = (x \cap y) \cup (x \cap z)$$
$$x \cup (y \cap z) = (x \cup y) \cap (x \cup z)$$

4.2 Boolesche Algebra

In einer **boolesche Algebra** gelten folgende Rechengesetze:

Satz 3 (cont.)

In der booleschen Algebra $(B, \cup, \cap, \bar{})$ für alle $x, y, z \in B$:

Neutralelement:
$$x \cup 0 = x$$
$$x \cap 1 = x$$

Nullelement:
$$x \cup 1 = 1$$
$$x \cap 0 = 0$$

Idempotenz:
$$x = x \cup x$$
$$x = x \cap x$$

Involution:
$$\mathrm{x} = \bar{\bar{x}}$$

4.2 Boolesche Algebra

In einer **boolesche Algebra** gelten folgende Rechengesetze:

Satz 3 (cont.)

In der booleschen Algebra $(B, \cup, \cap, \bar{\ })$ für alle $x, y, z \in B$:

Absorption:

$$(x \cup y) \cap x = x$$
$$(x \cap y) \cup x = x$$

Resolution:

$$(x \cup y) \cap (\bar{x} \cup y) = y$$
$$(x \cap y) \cup (\bar{x} \cap y) = y$$

Komplementarität:

$$x \cup (y \cap \bar{y}) = x$$
$$x \cap (y \cup \bar{y}) = x$$

de Morgan:

$$\overline{x \cup y} = \bar{x} \cap \bar{y}$$
$$\overline{x \cap y} = \bar{x} \cup \bar{y}$$

4.2 Boolesche Algebra

Wie beweist man einzelne Rechengesetze?

Wir stellen eine Wertetabelle auf.

Beispiel: **Absorption** $(x \cup y) \cap \ x = x$

x	y	$x \cup y$	linke Seite $(x \cup y) \cap \ x$	rechte Seite x	
0	0	0	0	0	✓
0	1	1	0	0	✓
1	0	1	1	1	✓
1	1	1	1	1	✓

4. Boolesche Funktionen und Schaltnetze

1. Einleitung ✓
2. Boolesche Algebra ✓
3. **Repräsentationen boolescher Funktionen**
4. Normalformen boolescher Funktionen
5. Repräsentation boolescher Funktionen mit OBDDs
6. Schaltnetze

4.3 Repräsentationen boolescher Funktionen

Wir benötigen im Folgenden den Begriff der booleschen Funktion.

Definition 4

Seien $n, m \in \mathbb{N}$ und B eine boolesche Algebra. Eine Funktion f : $B^n \rightarrow B^m$ heißt **boolesche Funktion**.

Notation

- B^n= Menge aller n-stelligen Tupel über B
- Beispiel $B^1 = B$= {(0), (1)}
- Beispiel B^2= {(0, 0), (0, 1), (1, 0), (1, 1)}
- Beispiel B^3= {(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)}

4.3 Repräsentationen boolescher Funktionen

Anzahl boolescher Funktionen

- boolesche Funktion f : $B^n \rightarrow B^m$ als Wertetabelle darstellbar, mit
- $|B^n| = 2^n$ Zeilen
- $|B^m| = 2^m$ Funktionswerten pro Zeilen

→$2^{m^{2^n}} = 2^{m \cdot 2^n}$ boolesche Funktionen für f : $B^n \rightarrow B^m$

4.3 Repräsentationen boolescher Funktionen

Grundfunktion aus dem Bereich der booleschen Funktionen

- boolesche Grundfunktion $f: B^n \to B^1$ als Wertetabelle darstellbar, mit
- $|B^n| = 2^n$ Zeilen
- $|B^1| = 2^1$ 1 Funktionswert pro Zeilen

Beispiel 1-stellige boolesche Grundfunktionen

- $B^1 \to B^1$
- $2^{1 \cdot 2^1} = 4$

	x = 0	x = 1	Funktion	Name
f_0	0	0	Konstante 0	*Kontradiktion*
f_1	0	1	x	*Identität*
f_2	1	0	$\bar{x}$	*Negation*
f_3	1	1	Konstante 1	*Tautologie*

4.3 Repräsentationen boolescher Funktionen

Alle booleschen 2-stelligen Grundfunktionen $B^2 \to B^1 (2^{1 \cdot 2^2} = 16)$

x_1, x_2	0, 0	0, 1	1, 0	1, 1	Funktion	Name
f_0	0	0	0	0	*0*	*Kontradiktion*
f_1	0	0	0	1	$\wedge$	*Konjunktion*
f_2	0	0	1	0		*Inhibition von x_1*
f_3	0	0	1	1	x_1	*Identität von x_1*
f_4	0	1	0	0		*Inhibition von x_2*
f_5	0	1	0	1	x_2	*Identität von x_2*
f_6	0	1	1	0	$\oplus$	*Antivalenz, Alternative*
f_7	0	1	1	1	$\vee$	*Disjunktion*
f_8	1	0	0	0	$\overline{\vee}$	*Nihilition*
f_9	1	0	0	1	$\Leftrightarrow$	*Äquivalenz*
f_{10}	1	0	1	0	$\overline{x_2}$	*Negation von x_2*
f_{11}	1	0	1	1		*Replikation*
f_{12}	1	1	0	0	$\overline{x_1}$	*Negation von x_1*
f_{13}	1	1	0	1	$\Rightarrow$	*Implikation*
f_{14}	1	1	1	0	$\overline{\wedge}$	*Exklusion*
f_{15}	1	1	1	1	*1*	*Tautologie*

4.3 Repräsentationen boolescher Funktionen

Im Folgenden werden diese Symbole verwenden:

- Konjunktion (und) $\wedge$
- Disjunktion (oder) $\vee$
- Negation (nicht) $\bar{\ }$

- Antivalenz (xor) $\oplus$

In der Regel abkürzende Notation der Konjunktion, z.B.: $\overline{x}_1 \wedge x_2$ → $\overline{x}_1 x_2$

Feste Folge der Funktionswerte, entsprechend des Wertes der Binärzahl:

Beispiel: f_6=(0,1,1,0) korrespondiert zu

x	y	f_6
0	0	0
0	1	1
1	0	1
1	1	0

4.3 Repräsentationen boolescher Funktionen

Vorsicht bei der Notation

- $\overline{x_1 x_2} \neq \overline{x}_1 \overline{x}_2$

x_1	x_2	$\overline{x_1 x_2}$	$\overline{x}_1 \overline{x}_2$
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

4.4 Normalformen boolescher Funktionen

Einschlägige und nicht einschlägige Indizes

- Wir tragen die Werte der Argumente gemäß ihrer natürlichen Ordnung (**Index**) in eine Wertetabelle ein
- Ist der Funktionswert für einen Index = 1 , nennen wir den Index **einschlägig**,
- Ist der Funktionswert für einen Index = 0 , nennen wir den Index **nicht einschlägig**

Beispiel:

Index	x_1	x_2	f_6	
0	0	0	0	nicht einschlägig
1	0	1	1	einschlägig
2	1	0	1	einschlägig
3	1	1	0	nicht einschlägig

4.4 Normalformen boolescher Funktionen

Minterm

Definition 5

Die boolesche Funktion m_i, für die nur der Index i einschlägig ist, heißt **Minterm** zum Index i.

Ein **Minterm** ist nur mit Negationen und Konjunktionen darstellbar:

- ist die Eingangsbelegung an der Stelle $x_j = 0$, setzen wir $\overline{x}_j$
- ist die Eingangsbelegung an der Stelle $x_j = 1$, setzen wir x_j
- und bilden dann die Konjunktion der Literale

Literal: In der mathematischen Logik ist ein **Literal** eine atomare Aussage (Atom) oder die Negation einer atomaren Aussage. Hier also eine Variable oder die negierte Variable.

4.4 Normalformen boolescher Funktionen

Beispiel zu Index und Minterm

Index	x_1	x_2	f_6	
0	0	0	0	nicht einschlägig
1	0	1	1	einschlägig
2	1	0	1	einschlägig
3	1	1	0	nicht einschlägig

Minterm zum Index 2 = $(10)_2$:

$$m_2(x_1, x_2) = x_1 \wedge \overline{x}_2$$

Was bewirkt nun m_2?

m_2 nimmt nur für den Index 2 den Wert 1 an.

Index	x_1	x_2	$x_1 \wedge \overline{x}_2$
0	0	0	0
1	0	1	0
2	1	0	1
3	1	1	0

4.4 Normalformen boolescher Funktionen

Hinführung zu Normalformen

Für wie viele Eingaben liefert ein Minterm 1?

Wie soeben gesehen für genau 1 Eingabe.

Folgerungen

- **Disjunktion** aller **Minterme** zu einschlägigen Indizes einer booleschen Funktion f ist wieder f
- **XOR-Verknüpfung** aller **Minterme** zu einschlägigen Indizes einer booleschen Funktion f ist wieder f

4.4 Normalformen boolescher Funktionen

Normalformen

Definition 8

Die Darstellung einer booleschen Funktion f als Disjunktion all ihrer Minterme zu einschlägigen Indizes heißt **disjunktive Normalform (DNF).**

Die Darstellung einer booleschen Funktion f als XOR-Verknüpfung all ihrer Minterme zu einschlägigen Indizes heißt **Ringsummen-Normalform (RNF).**

Anmerkung: Normalformen sind **eindeutig**.

Index	x_1	x_2	f_6	Minterme
0	0	0	0	$\overline{x}_1 \wedge \overline{x}_2$
1	0	1	1	$\overline{x}_1 \wedge x_2$
2	1	0	1	$x_1 \wedge \overline{x}_2$
3	1	1	0	$x_1 \wedge x_2$

Beispiel f_6

- DNF von f_6: $\overline{x}_1 x_2 \vee x_1 \overline{x}_2$
- RNF von f_6: $\overline{x}_1 x_2 \oplus x_1 \overline{x}_2$

4.4 Normalformen boolescher Funktionen

Funktionale Vollständigkeit

Beobachtung:

- jede boolesche Funktion $f : B^n \to B$ ist durch ausschließliche Verwendung von Konjunktion, Disjunktion und Negation darstellbar
- z. B. durch ihre DNF

Definition 8

Eine Menge F von booleschen Funktionen heißt **funktional vollständig**, wenn sich jede boolesche Funktion durch Einsetzen und Komposition von Funktionen aus F darstellen lässt.

Satz 6

$\mathrm{F} = \{\wedge, \vee, \overline{}\}$ ist **funktional vollständig**.

4.4 Normalformen boolescher Funktionen

Funktionale Vollständigkeit

Frage: Gibt es kleinere funktional vollständige Mengen F ?

Behauptung: Funktional vollständig sind

- $\mathrm{F}_1 = \left\{\wedge, \overline{}\right\}$
- $\mathrm{F}_2 = \left\{\vee, \overline{}\right\}$

Zum Beweis genügt es zu zeigen, dass $\left\{\wedge, \vee, \overline{}\right\}$ darstellbar ist.

Beweis (durch die de Morgan'schen Gesetze)

4.4 Normalformen boolescher Funktionen

Funktionale Vollständigkeit

Frage: Gibt es kleinere funktional vollständige Mengen F ?

Behauptung: Funktional vollständig sind

- $\mathrm{F}_1 = \left\{\wedge, \overline{}\right\}$
- $\mathrm{F}_2 = \left\{\vee, \overline{}\right\}$

Zum Beweis genügt es zu zeigen, dass $\left\{\wedge, \vee, \overline{}\right\}$ darstellbar ist.

Beweis

Anwendung der de Morgan'schen Gesetze (Satz 3):

- $x \vee y = \overline{\overline{x} \wedge \overline{y}}$ → $\{\vee\}$ ist durch $= \left\{\wedge, \overline{}\right\}$ darstellbar
- $x \wedge y = \overline{\overline{x} \vee \overline{y}}$ → $\{\wedge\}$ ist durch $= \left\{\vee, \overline{}\right\}$ darstellbar $\square$

4.4 Normalformen boolescher Funktionen

Funktionale Vollständigkeit

Frage: Gibt es noch kleinere funktional vollständige Mengen F ?

Behauptung: Funktional vollständig ist

- $\mathrm{F}_3 = \{NAND\}$
- $\mathrm{F}_4 = \{NOR\}$

Zum Beweis der funktionalen Vollständigkeit von F_3 genügt es zu zeigen, dass $\left\{\vee, \overline{}\right\}$ darstellbar ist.

Beweis Teil 1

Darstellung der Negation

- $\overline{x} = NAND(x, x)$

x	$\overline{x}$	$NAND(x, x)$
0	1	1
1	0	0

4.4 Normalformen boolescher Funktionen

Beweis Teil 2

Darstellung der Disjunktion

- $x \vee y \; = NAND(NAND(x,x), NAND(y,y))$

x	y	$x \vee y$	$\overline{x}$	$\overline{y}$	$NAND(NAND(x,x), NAND(y,y))$
0	0	0	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	1	0	0	1

$\square$

4.4 Normalformen boolescher Funktionen

Maxterme

Beobachtung: Minterm-Darstellung betont Funktionswert 1.

Definition

Die boolesche Funktion M_i, für die nur der Index i nicht einschlägig ist, heißt **Maxterm** zum Index i .

Beobachtungen

- Definition Maxterm unterscheidet sich nur in "nicht" von Definition Minterm
- wenn m_i Minterm zum Index i und M_i Maxterm zum Index i ist

$$\Rightarrow M_i = \overline{m_i}$$

- Konjunktion aller Maxterme zu nicht einschlägigen Indizes einer booleschen Funktion f ist wieder f

4.4 Normalformen boolescher Funktionen

Normalformen

Fortsetzung von Definition 8

Die Darstellung von f als Konjunktion all ihrer Maxterme zu nicht einschlägigen Indizes heißt **konjunktive Normalform (KNF).**

Beispiel $f_{bsp} = (x \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z})$

Index	x	y	z	f_{bsp}	$x \vee y \vee \overline{z}$	$x \vee \overline{y} \vee z$	$x \vee \overline{y} \vee \overline{z}$
0	0	0	0	1	1	1	1
1	0	0	1	**0**	**0**	1	1
2	0	1	0	**0**	1	**0**	1
3	0	1	1	**0**	1	1	**0**
4	1	0	0	1	1	1	1
5	1	0	1	1	1	1	1
6	1	1	0	1	1	1	1
7	1	1	1	1	1	1	1

4.4 Normalformen boolescher Funktionen

Wozu stellt man boolesche Funktionen dar?

- Realisierung
- Verifikation
- Fehleranalyse
- Synthese
- . . .

Wo stellt man boolesche Funktionen dar?

- auf dem Papier
- im Computer

Probleme

- Wertetabelle, Wertevektor immer groß
- Normalformen oft groß
- Normalformen unterstützen gewünschte Operationen kaum

Wunsch: andere Repräsentation

4. Boolesche Funktionen und Schaltnetze

1. Einleitung ✓
2. Boolesche Algebra ✓
3. Repräsentationen boolescher Funktionen ✓
4. Normalformen boolescher Funktionen ✓
5. **Repräsentation boolescher Funktionen mit OBDDs**
6. Schaltnetze

4.5 Ordered Binary Decision Diagrams

Eine Datenstruktur für boolesche Funktionen

Ziel: Darstellung für f : $B^n \rightarrow B$

Wünsche:

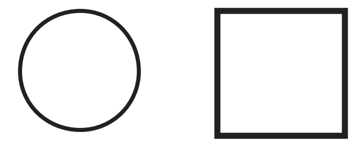
- zu einer Belegung $x_1, x_2, \ldots, x_n$ schnell den Funktionswert $f(x_1, x_2, \ldots, x_n)$ ausrechnen können
- Funktionen schnell auf Gleichheit testen können
- Funktionen schnell manipulieren (z. B. eine Variable konstant setzen) können
- schnell eine Null-Eingabe/eine Eins-Eingabe finden können
- Funktionen möglichst klein repräsentieren
- . . .

Ordered **B**inary **D**ecision **D**iagrams

4.5 Ordered Binary Decision Diagrams

OBDDs

- erster Schritt; Festlegung einer **Variablenordnung** π
  z. B.: $\pi = (x_3, x_1, x_2, x_4)$ bei einer 4-stelligen Funktion
- zweiter Schritt: OBDD bauen
  - aus **Knoten** für Variablen oder Konstanten



  - **Kanten**, die zwei Knoten verbinden

  - nach folgenden Regeln:

Regeln

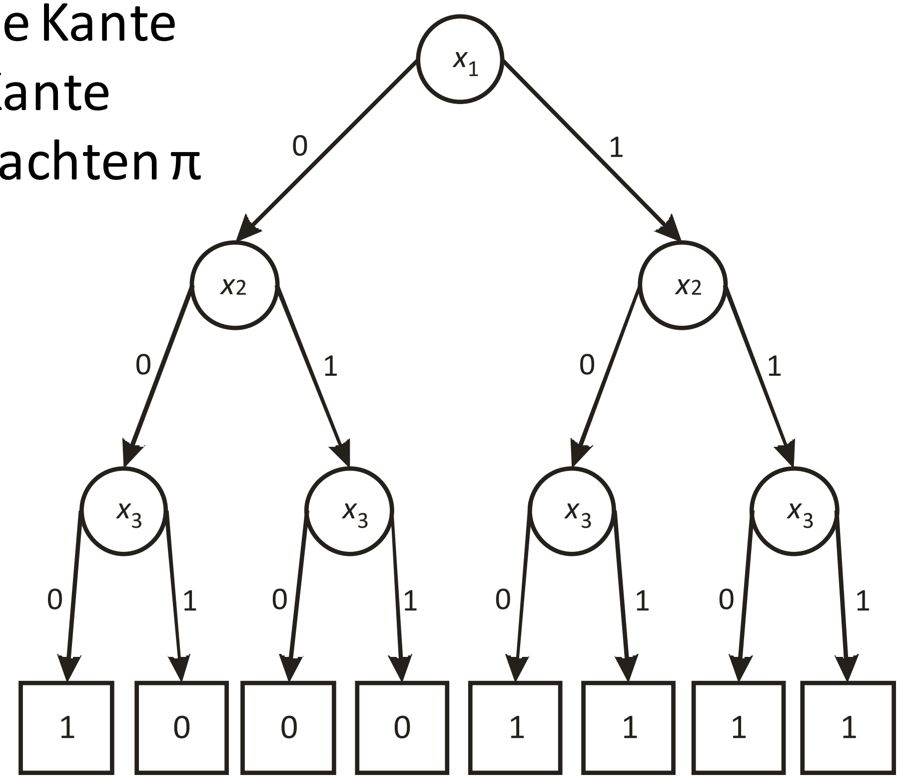
- **Knoten mit Variablen**, 0 oder 1 markiert
- **Kanten** mit 0 oder 1 markiert
- **Variablen-Knoten** mit je einer ausgehenden 0- und 1-Kante
- **Konstanten-Knoten** ohne ausgehende Kante
- genau ein Knoten ohne eingehende Kante
- Kanten zwischen Variablenknoten beachten π

4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Regeln

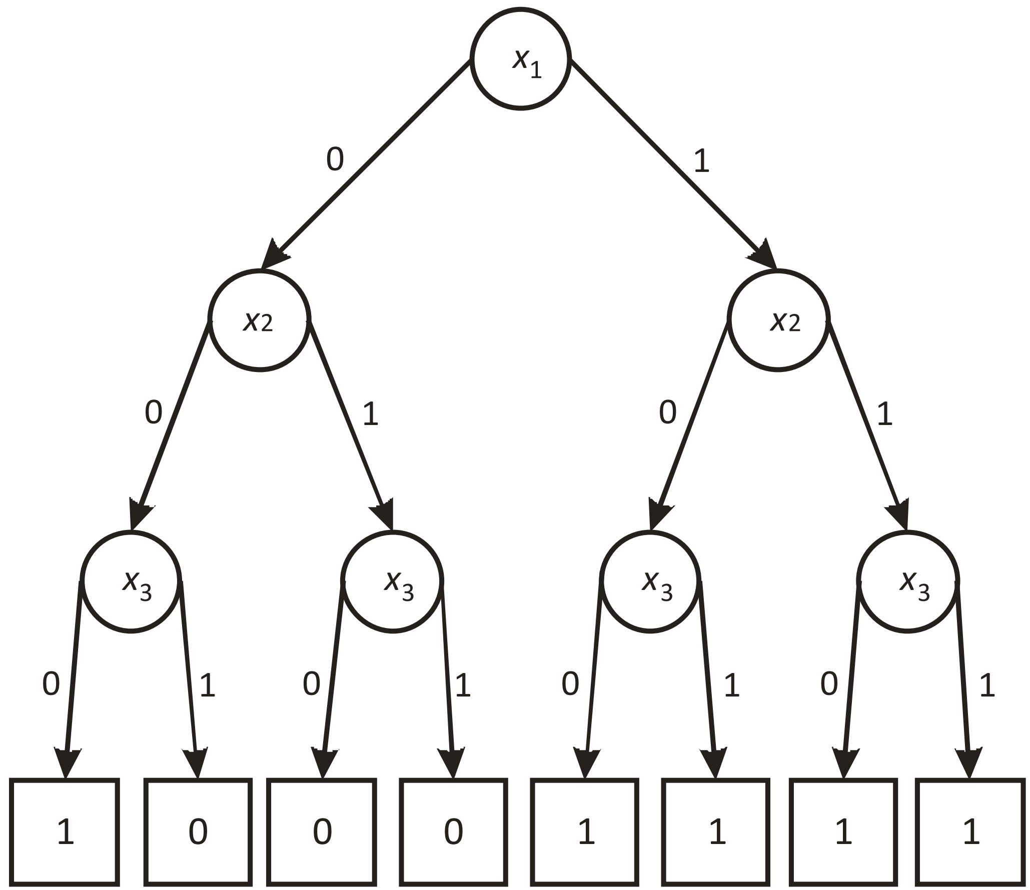
- **Knoten mit Variablen**
- **Kanten** mit 0 oder 1 markiert
- **Variablen-Knoten** mit je einer ausgehenden 0- und 1-Kante
- **Konstanten-Knoten** ohne ausgehende Kante
- genau ein Knoten ohne eingehende Kante
- Kanten zwischen Variablenknoten beachten π



4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Index	x_1	x_2	x_3	f_{bsp}
0	0	0	0	1
1	0	0	1	**0**
2	0	1	0	**0**
3	0	1	1	**0**
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



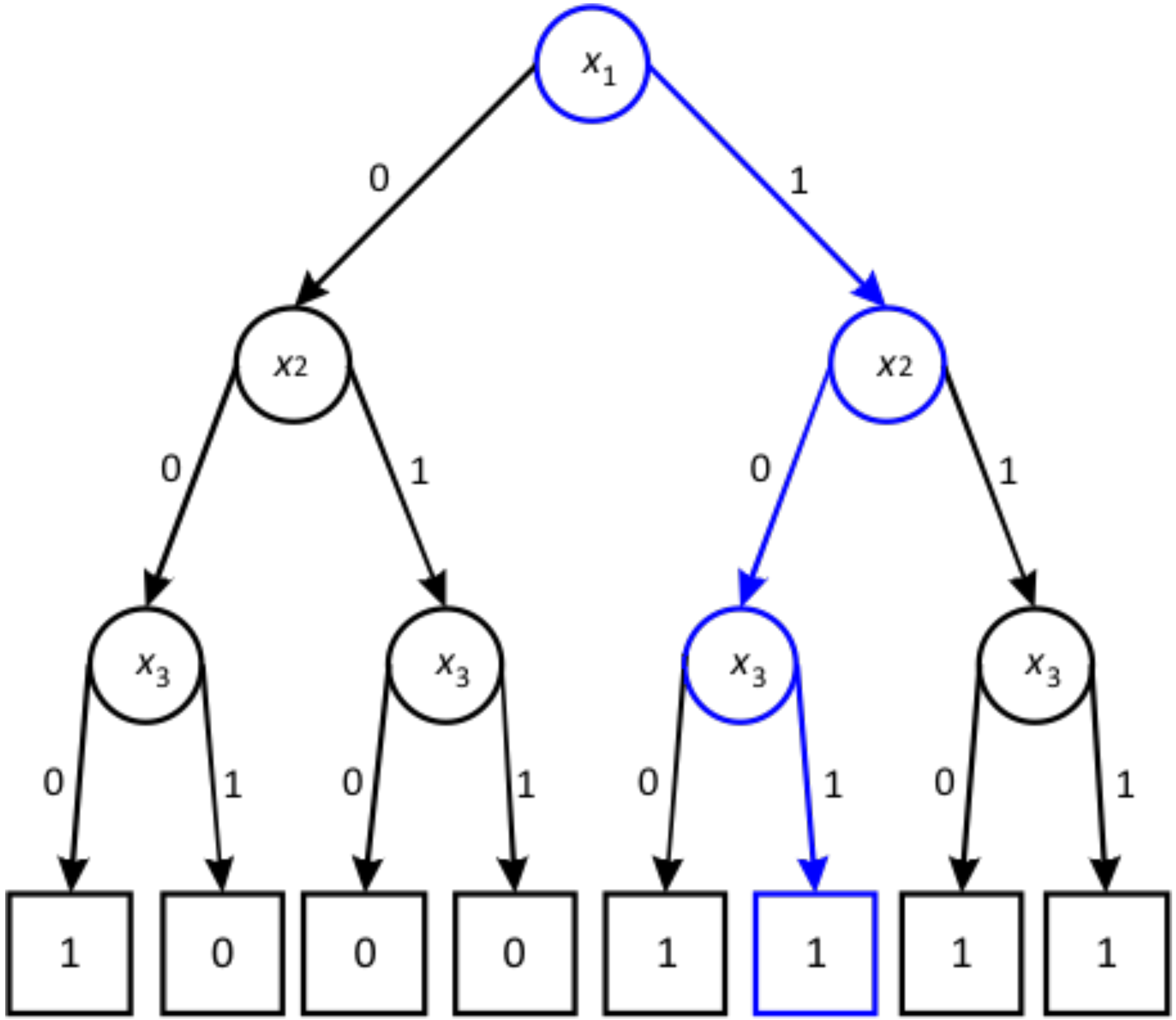
4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Auswertung $f(1, 0, 1)$

- $x_1 = 1$
- $x_2 = 0$
- $x_3 = 1$

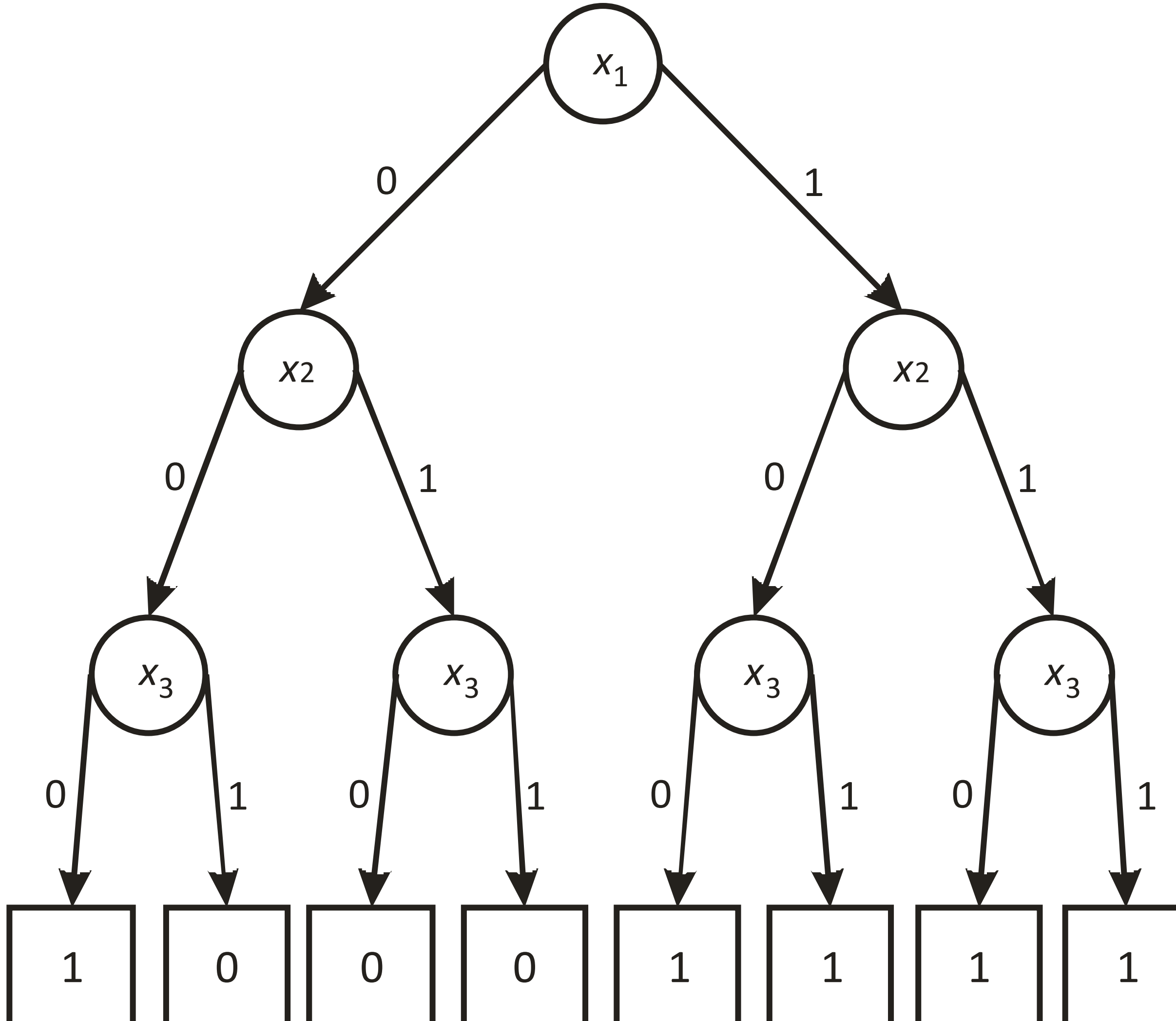
→ $f(1, 0, 1) = 1$



4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Ist das nicht etwas groß?

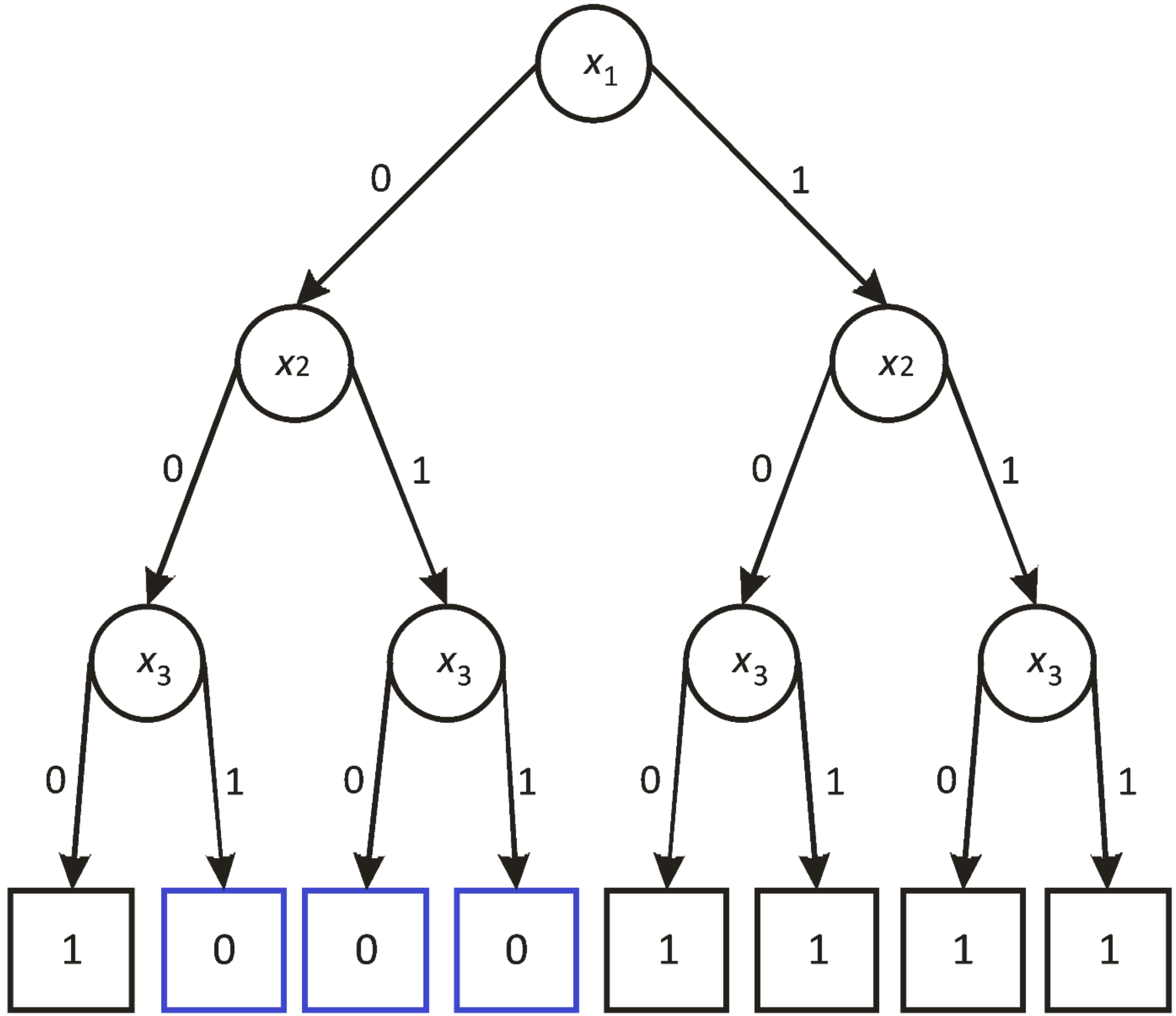


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen

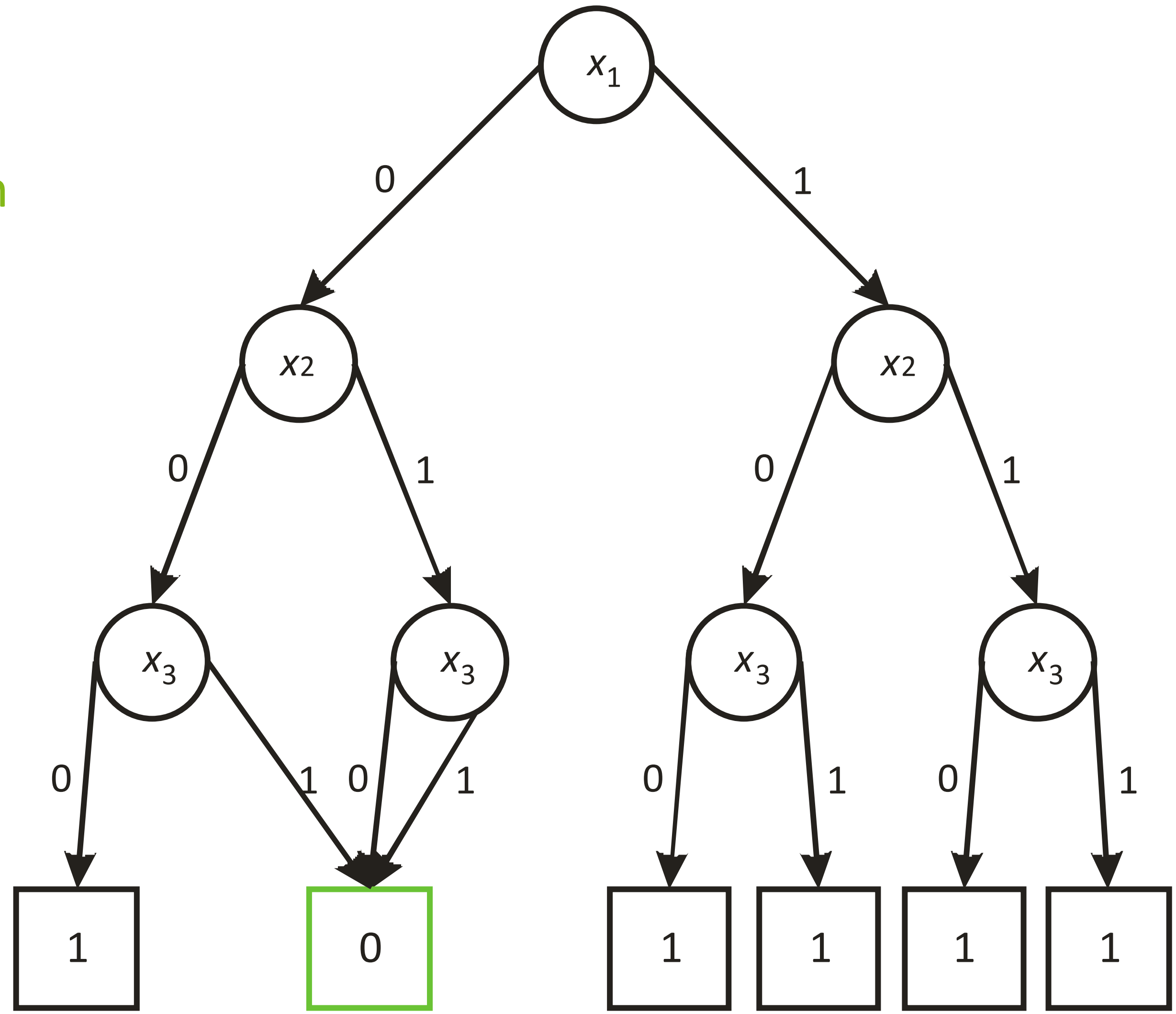


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen

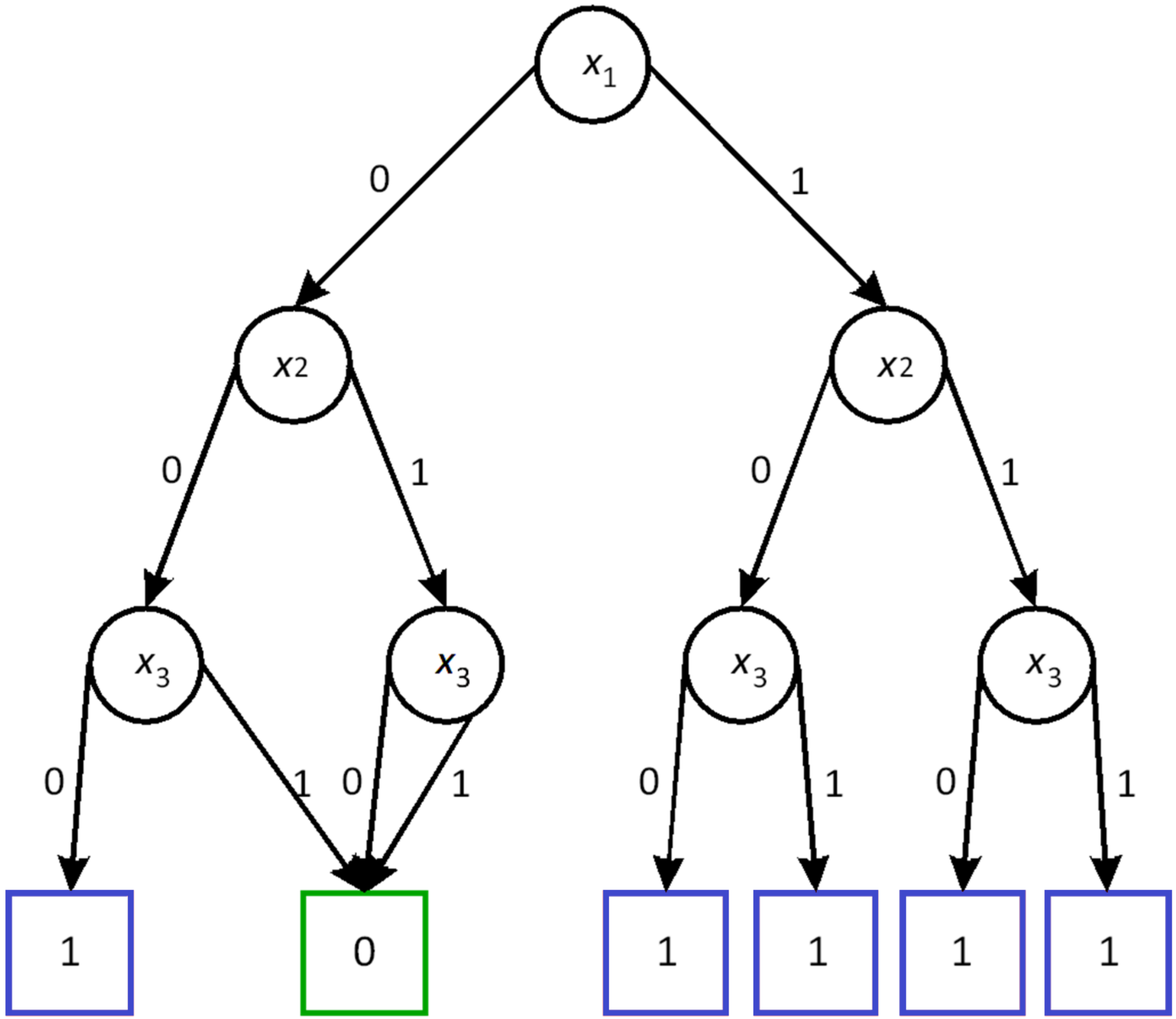


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen

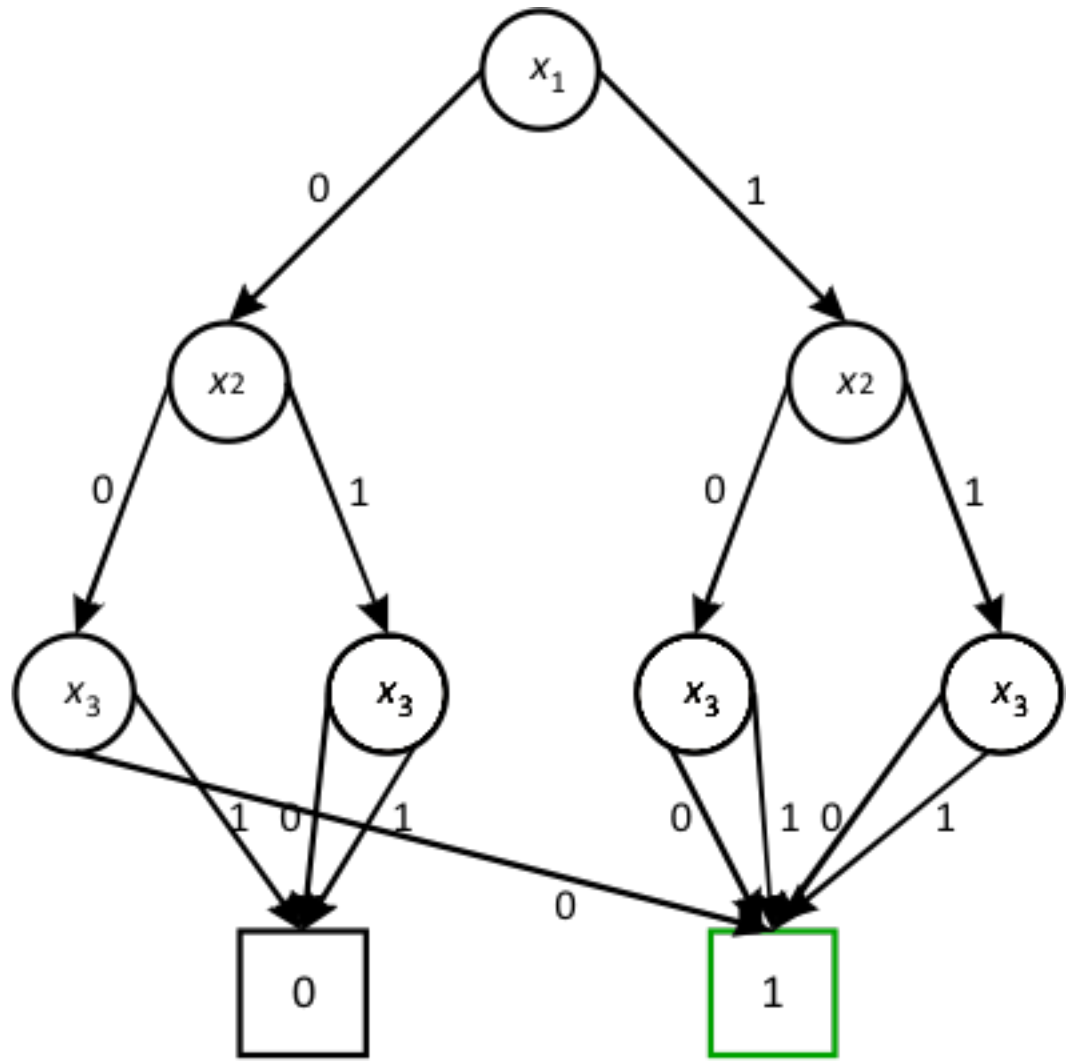


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

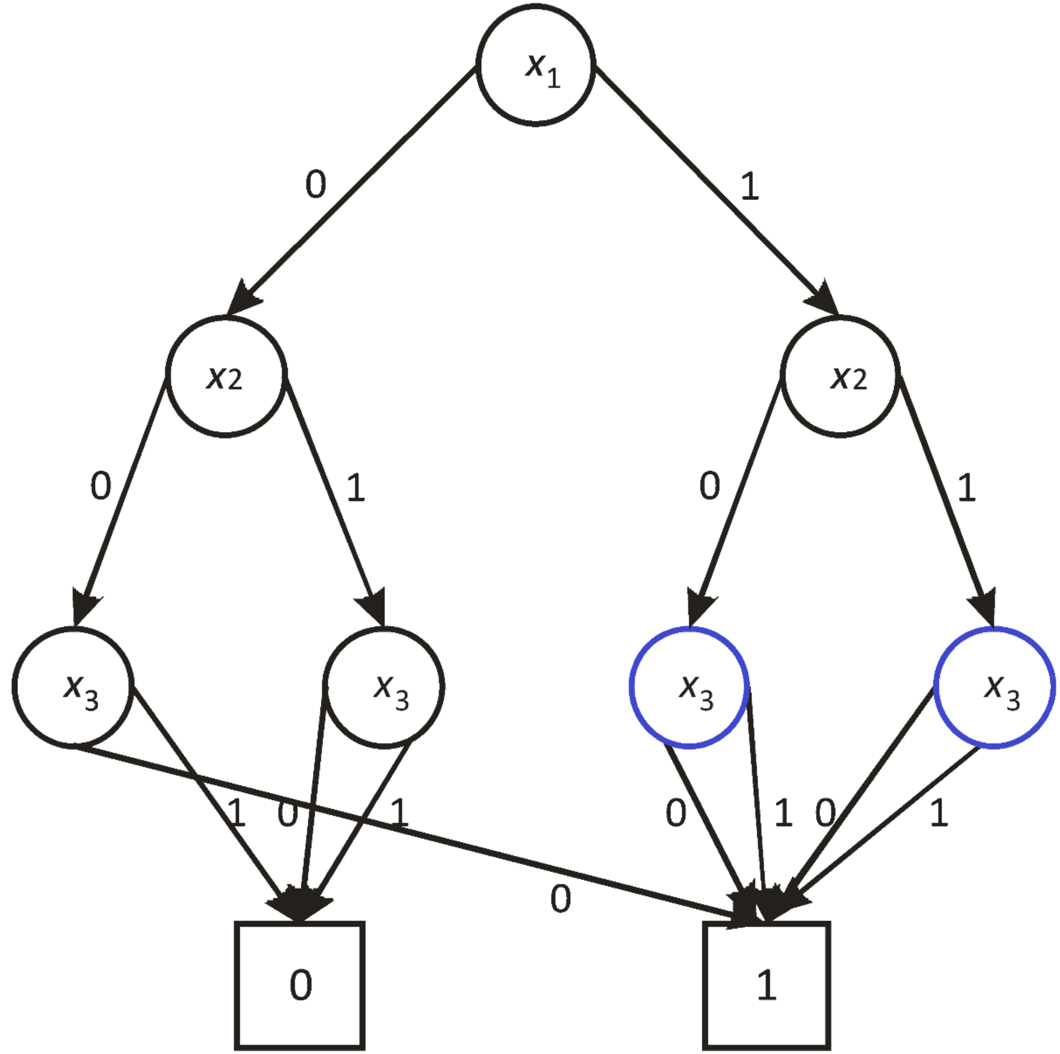
Vereinfachungen

- gleichartige Senken verschmelzen



4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$



Vereinfachungen

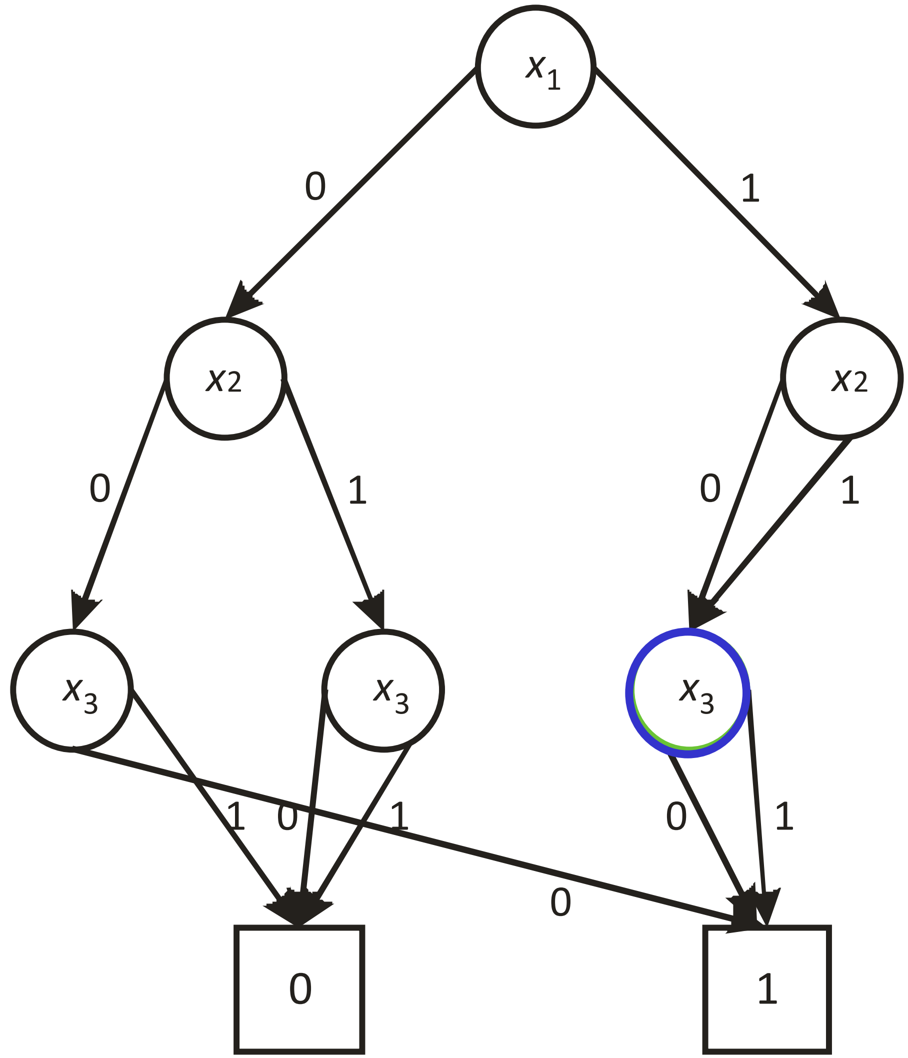
- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen

4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen

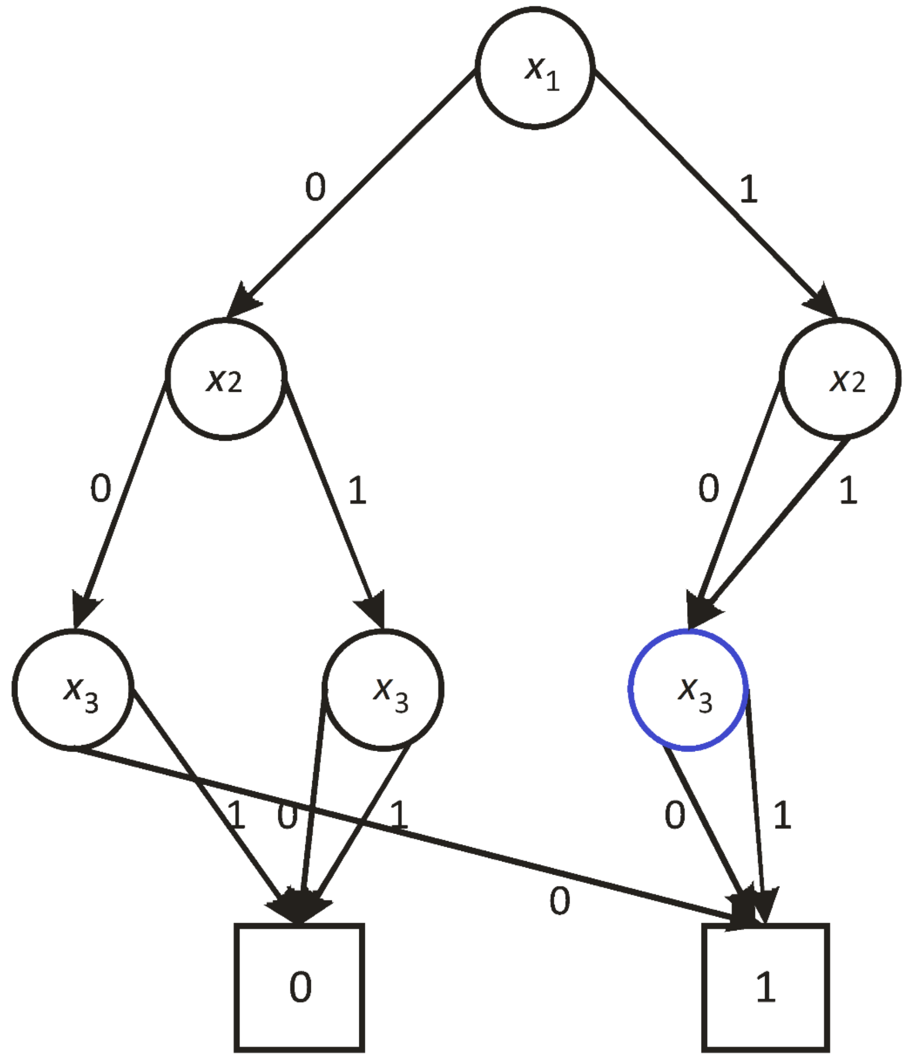


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen
- Knoten ohne Einfluss entfernen

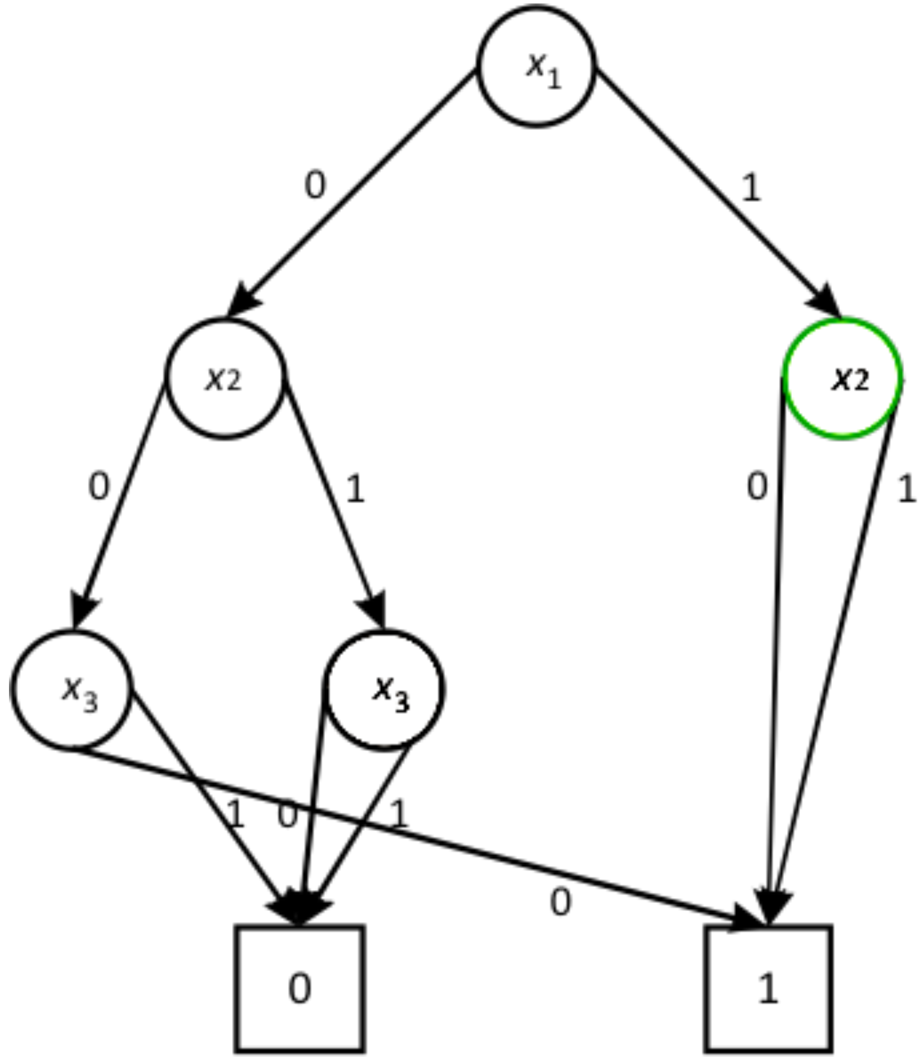


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen
- Knoten ohne Einfluss entfernen

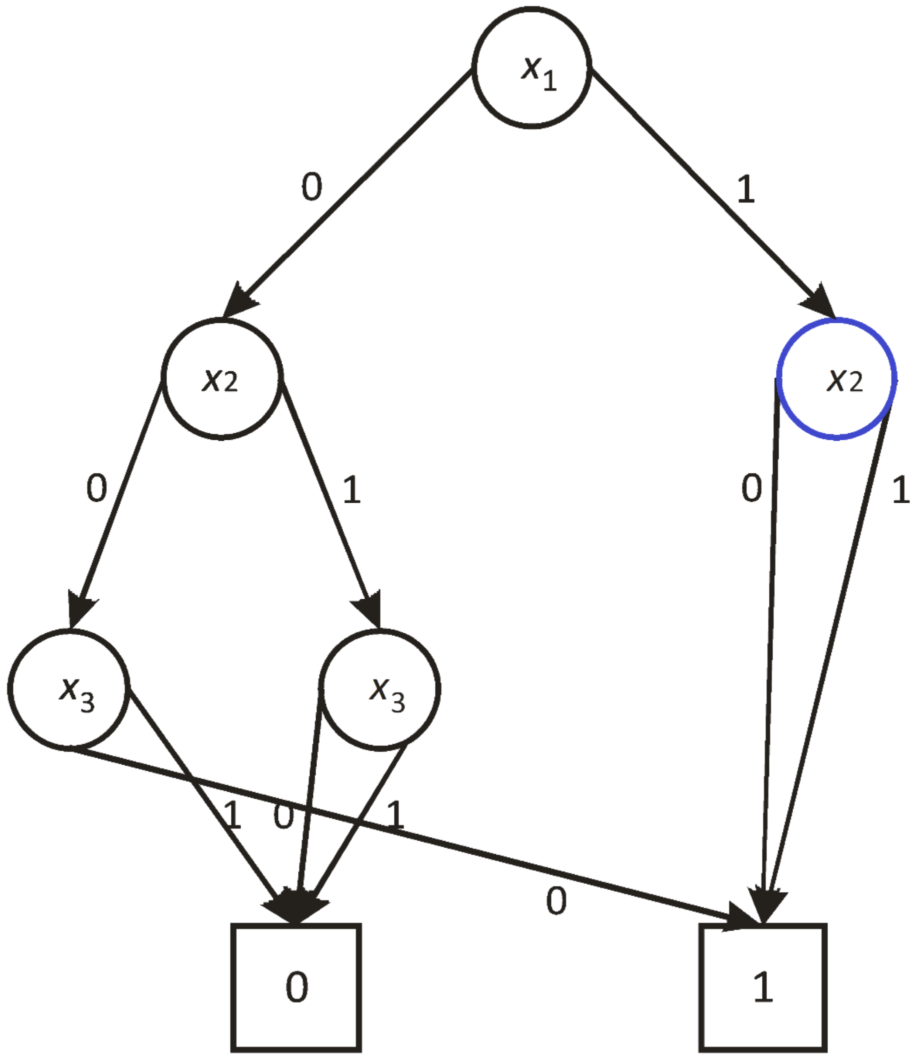


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen
- Knoten ohne Einfluss entfernen

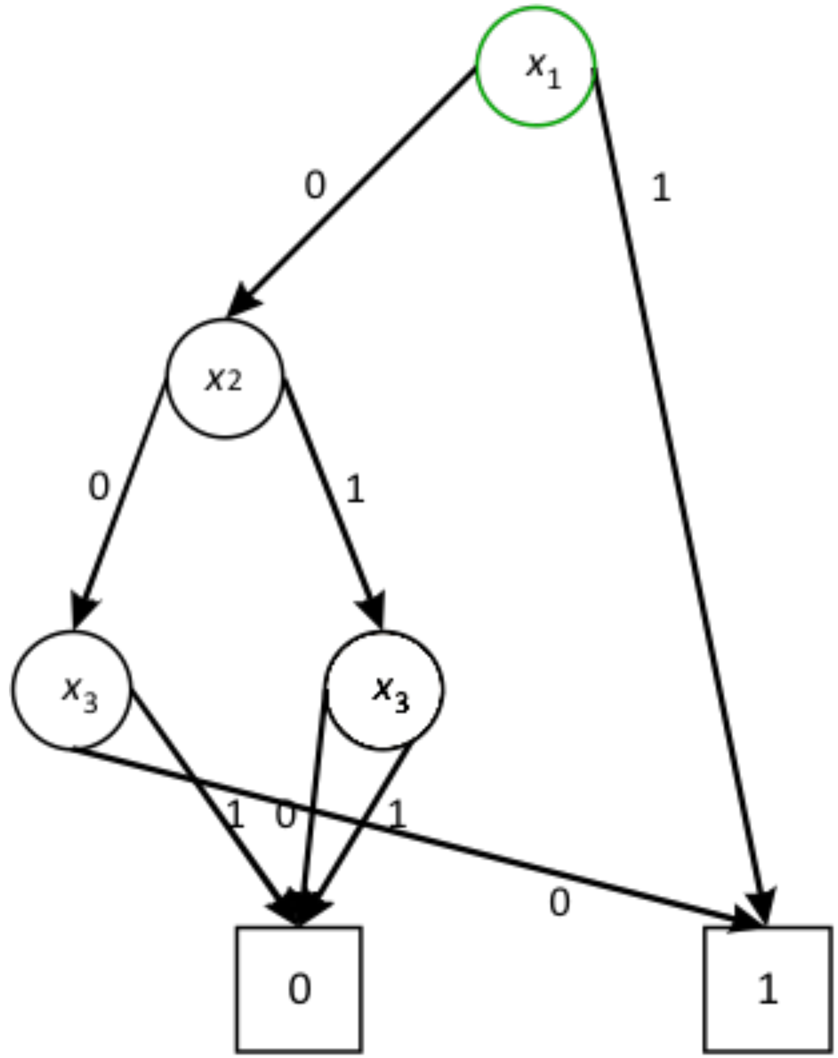


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen
- Knoten ohne Einfluss entfernen

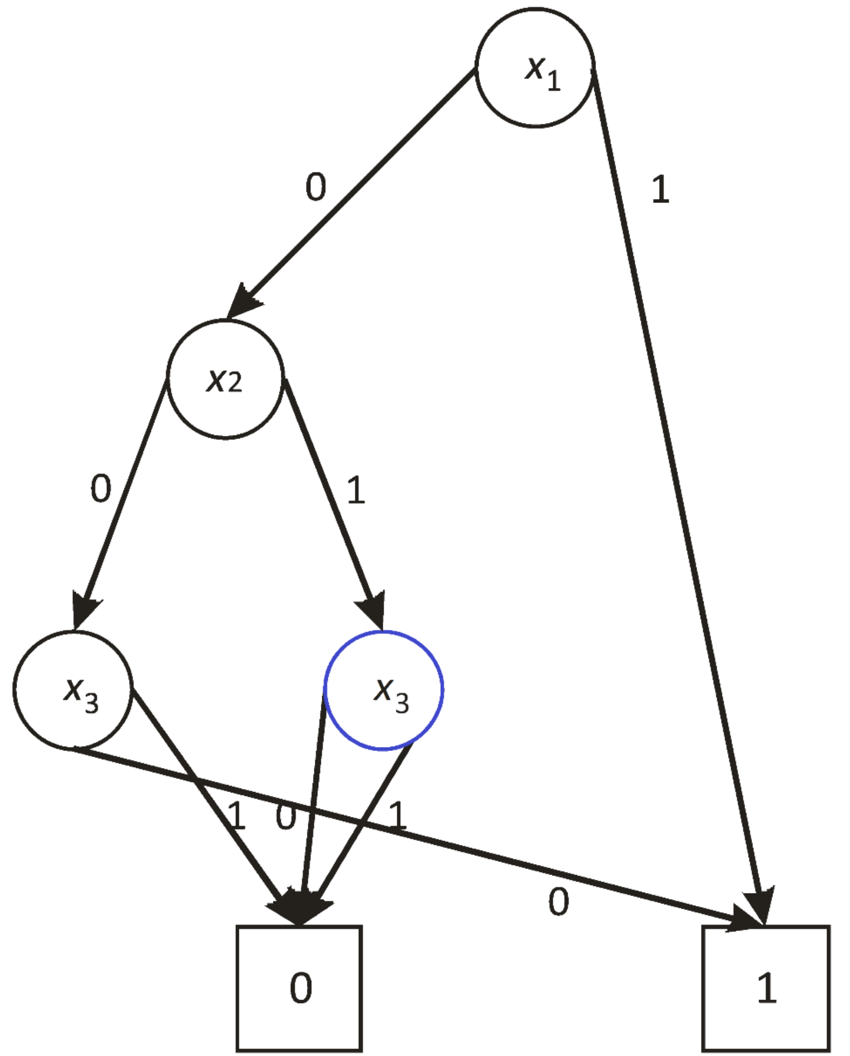


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen
- Knoten ohne Einfluss entfernen

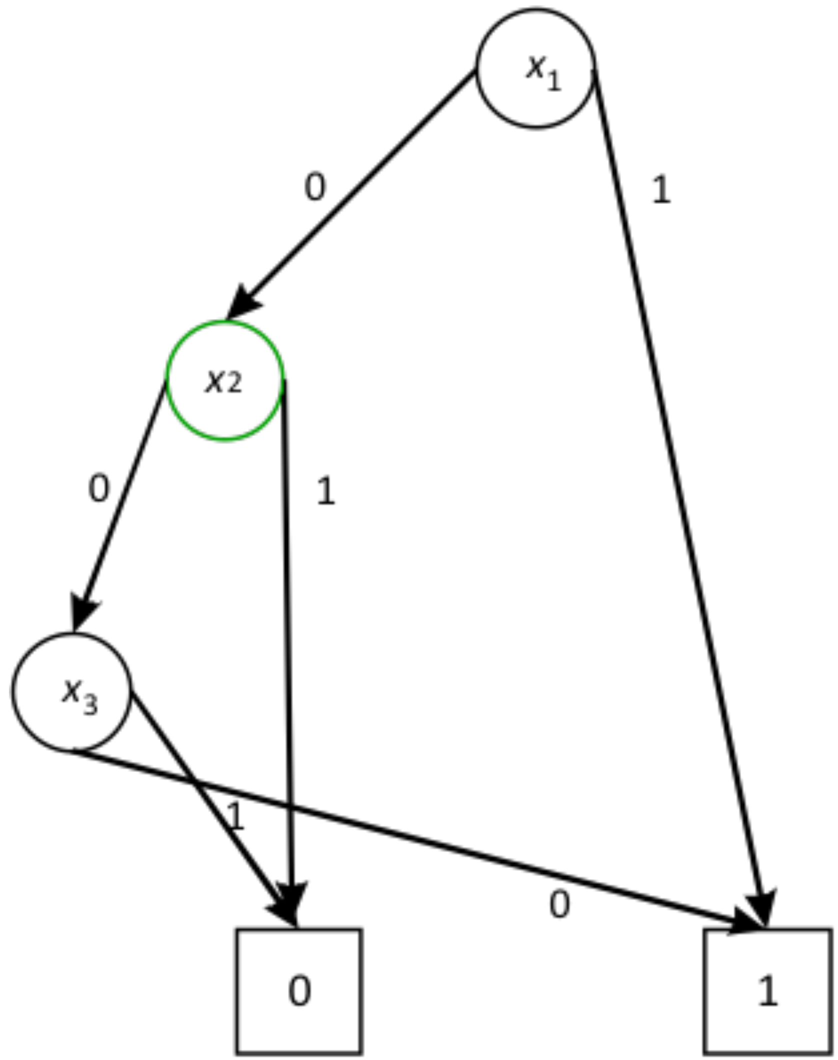


4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen
- Knoten ohne Einfluss entfernen



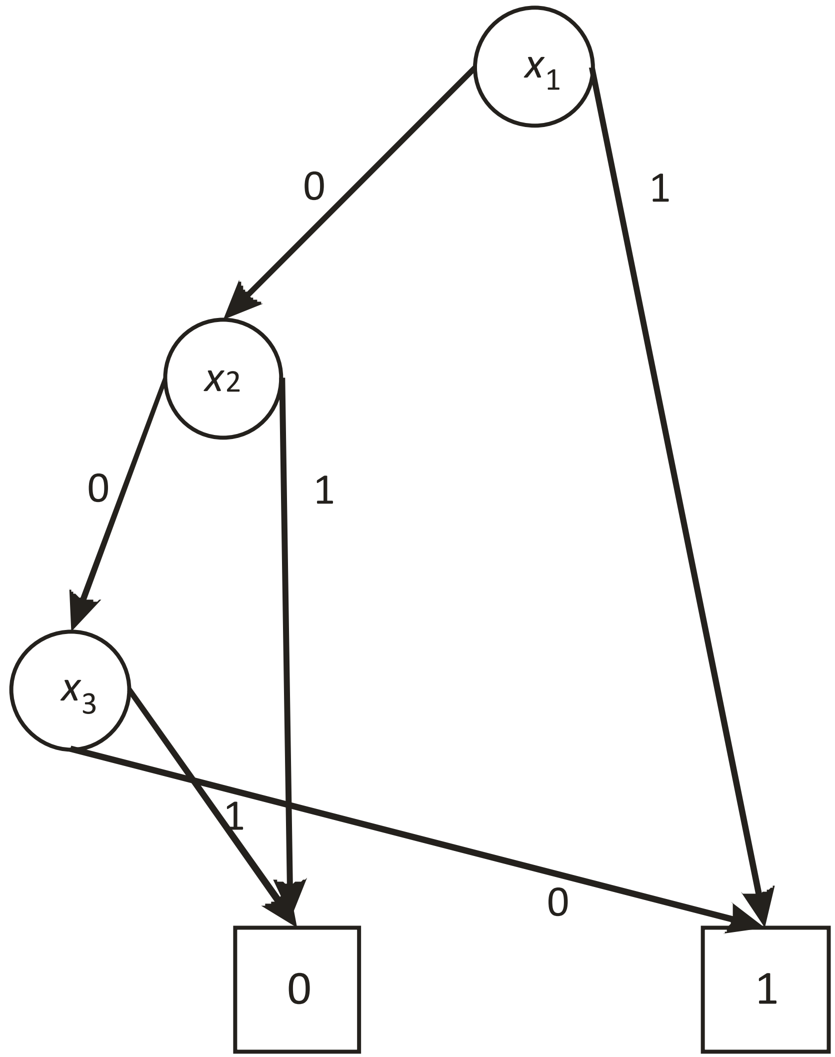
4.5 Ordered Binary Decision Diagrams

OBDD – Ein Beispiel $\pi = (x_1, x_2, x_3)$

Vereinfachungen

- gleichartige Senken verschmelzen
- gleichartige Knoten verschmelzen
- Knoten ohne Einfluss entfernen

→ OBDD minimal



OBDD-Reduzierung

Satz 9

Die erschöpfende Anwendung der

- Verschmelzungsregel
  Knoten mit gleicher Markierung und gleichen Nachfolgern können verschmolzen werden und der
- Eliminationsregel
  Ein Knoten mit gleichem Null- und Einsnachfolger kann entfernt werden

in beliebiger Reihenfolge führt zum **reduzierten** πOBDD.

reduziert bedeutet: OBDD hat minimale Größe und ist eindeutig definiert

4.5 Ordered Binary Decision Diagrams

Was bringt und die OBDD-Reduzierung

Originalfunktion als DNF

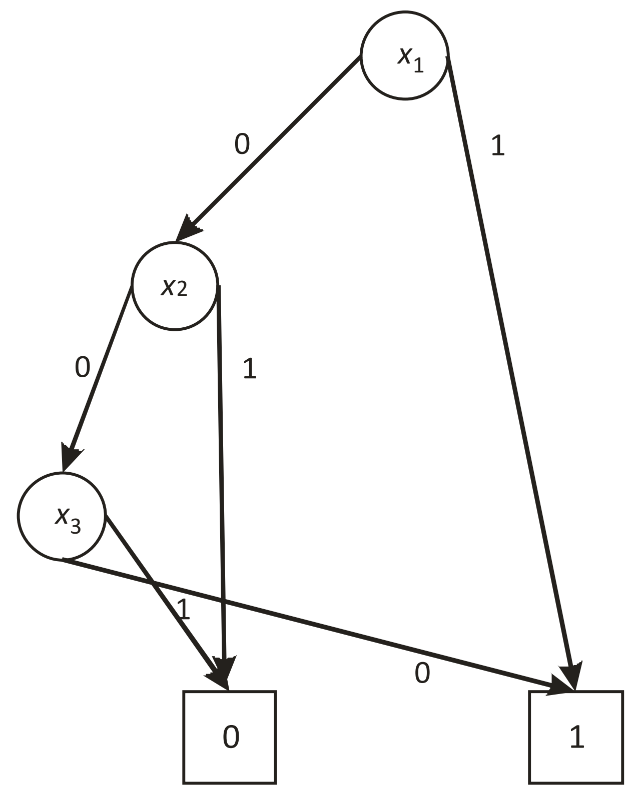
$$f_{bsp} = \overline{x}\,\overline{y}\,\overline{z} \vee x\,\overline{y}\,\overline{z} \vee x\,\overline{y}z \vee x\,y\,\overline{z} \vee x\,y\,z$$

Index	x	y	z	f_{bsp}
0	0	0	0	1
1	0	0	1	**0**
2	0	1	0	**0**
3	0	1	1	**0**
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

4.5 Ordered Binary Decision Diagrams

OBDD in boolesche Funktion umwandeln

- disjunktive Form, d. h. Disjunktion von Termen, die nur Negation und Konjunktion enthalten
- wir berücksichtigen nur Kanten, die zu Konstanten-Knoten mit 1 führen
- folgen wir für eine Variable x einer 1-Kante, setzen wir x als Literal in den Term
- folgen wir für eine Variable x einer 0-Kante, setzen wir $\overline{x}$ als Literal in den Term
- Literale werden mit der Konjunktion (UND) verknüpft
- jeder Pfad zu einem Konstanten-Knoten mit 1 ergibt einen Term
- alle entstandenen Terme werden mit der Disjunktion (ODER) verknüpft



→ hier: $f_{bsp} = \overline{x_1}\ \overline{x_2}\ \overline{x_3} \vee x_1$

4.5 Ordered Binary Decision Diagrams

Was bringt und die OBDD-Reduzierung

Originalfunktion als DNF

- $f_{bsp} = \overline{x}\,\overline{y}\,\overline{z} \lor x\,\overline{y}\,\overline{z} \lor x\,\overline{y}z \lor x\,y\,\overline{z} \lor x\,y\,z$

Originalfunktion aus OBDD-Reduzierung

- $f_{bsp} = \overline{x}\,\overline{y}\,\overline{z} \lor x$
- weniger Terme
- einfachere Terme

Index	x	y	z	f_{bsp}
0	0	0	0	1
1	0	0	1	**0**
2	0	1	0	**0**
3	0	1	1	**0**
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

4. Boolesche Funktionen und Schaltnetze

1. Einleitung ✓
2. Boolesche Algebra ✓
3. Repräsentationen boolescher Funktionen ✓
4. Normalformen boolescher Funktionen ✓
5. Repräsentation boolescher Funktionen mit OBDDs ✓
6. **Schaltnetze**

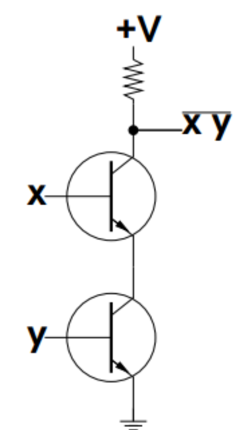
4.6 Schaltnetze

Wo bleibt die Hardware?

- bis jetzt haben wir über theoretische Grundlagen diskutiert
- wir betrachten die Hardware auf abstrakter Ebene

Wunsch

- Realisierung boolescher Funktionen in Hardware
- wir benötigen eine funktional vollständige Menge boolescher Funktionen
- Erinnerung: technische Realisierung von **NAND** reicht aus
- Beobachtung: folgende Schaltung mit Transistoren realisiert die NAND-Funktion



4.6 Schaltnetze

Logische Gatter

- Realisierung mit Transistoren . . . **für uns die falsche Ebene!**

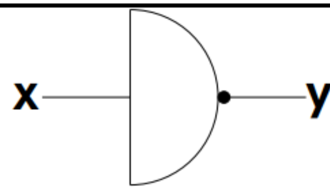
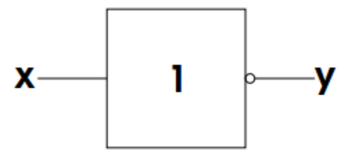
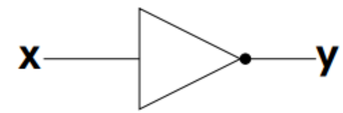
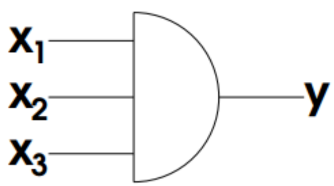
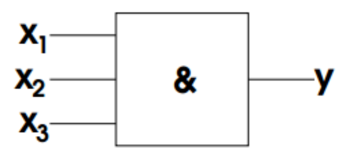
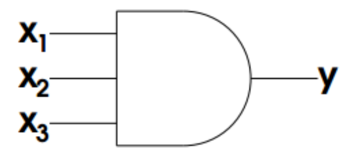
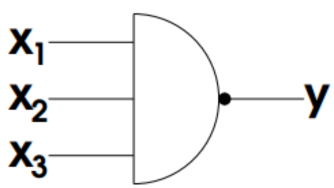
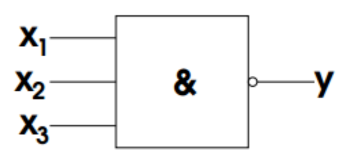
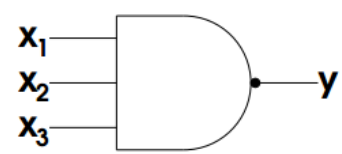
Grundlage für RS

- einfache logische Bausteine (logische Gatter)
- Bausteine für
  - Negation
  - Konjunktion
  - Disjunktion
- Regeln
  - Eingänge mit Variablen oder Konstanten belegt
  - nur Verbindungen von Ausgängen zu Eingängen
  - keine Kreise (Rückkopplung Ausgang → Eingang)

→ **Schaltnetz**

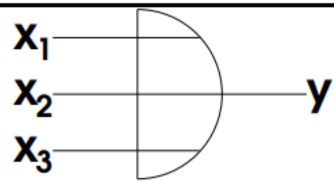
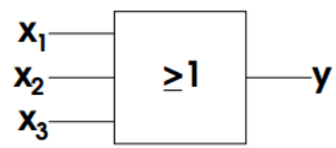
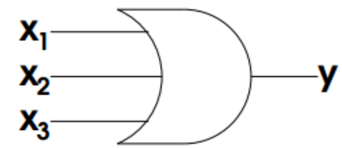
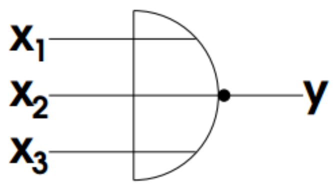
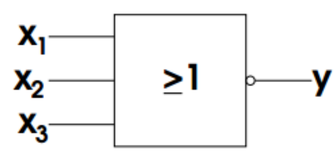
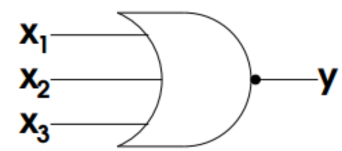
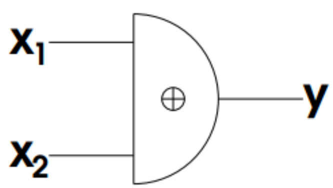
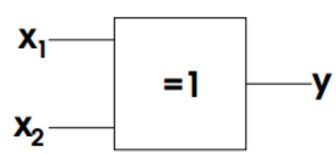
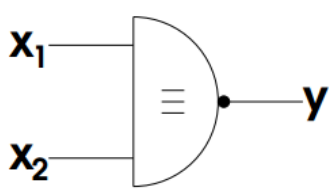
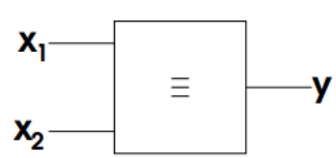
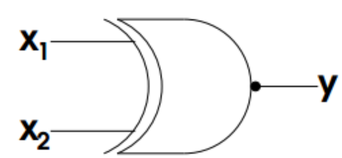
4.6 Schaltnetze

Symbole für logische Gatter (1)

Funktion	DIN 40700	DIN EN 60617	IEEE
$y = \overline{x}$			
$y = x_1 \wedge x_2 \wedge x_3$			
$y = \overline{x_1 \wedge x_2 \wedge x_3}$			

4.6 Schaltnetze

Symbole für logische Gatter (2)

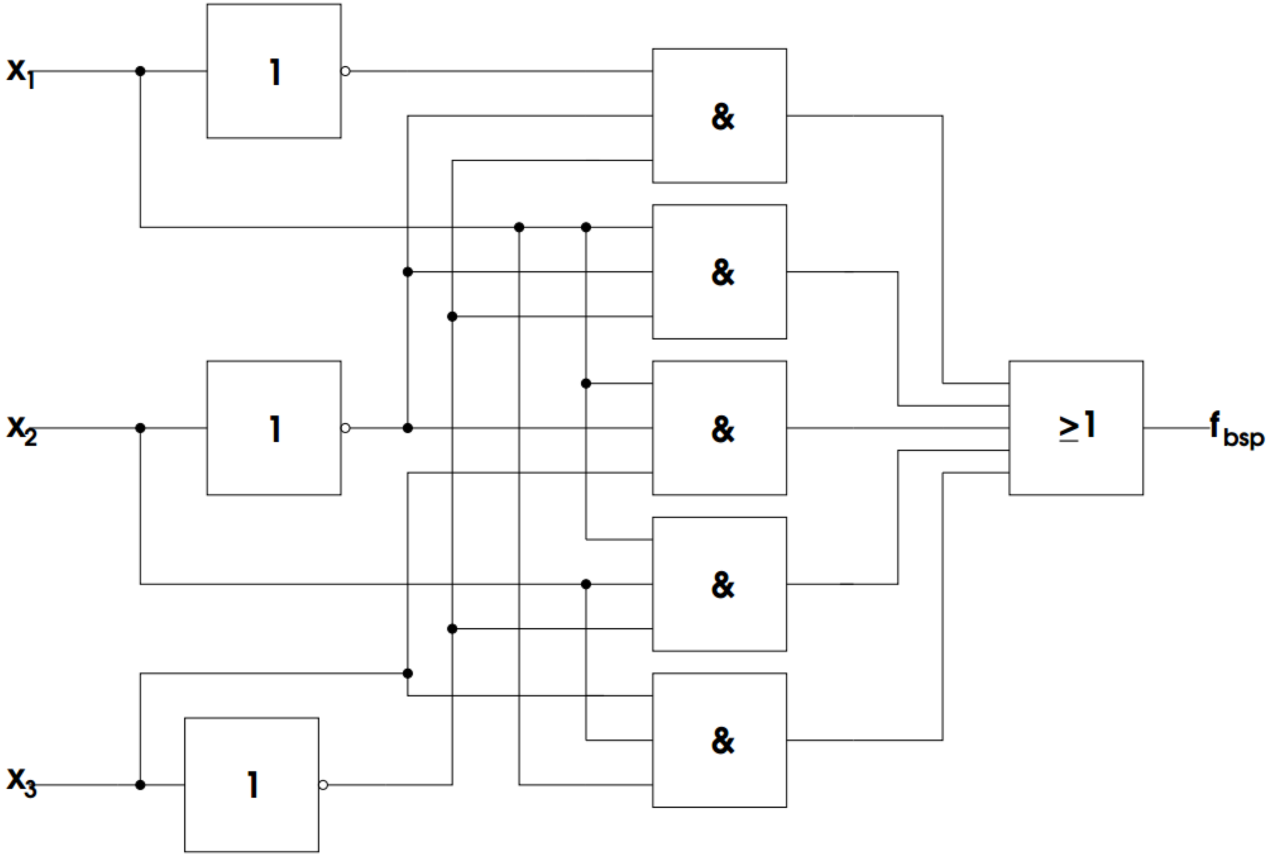
Funktion	DIN 40700	DIN EN 60617	IEEE
$y = x_1 \vee x_2 \vee x_3$			
$y = \overline{x_1 \vee x_2 \vee x_3}$			
$y = x_1 \oplus x_2$			
$y = \overline{x_1}\,\overline{x_2} \vee x_1\, x_2$			

4.6 Schaltnetze

Beispiel DNF

$f_{bsp}: B^3 \rightarrow B,$ Wertevektor (1, 0, 0, 0, 1, 1, 1, 1)

$$f_{bsp} = \overline{x}_1\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3 \vee x_1\,x_2\,\overline{x}_3 \vee x_1 x_2\,x_3$$

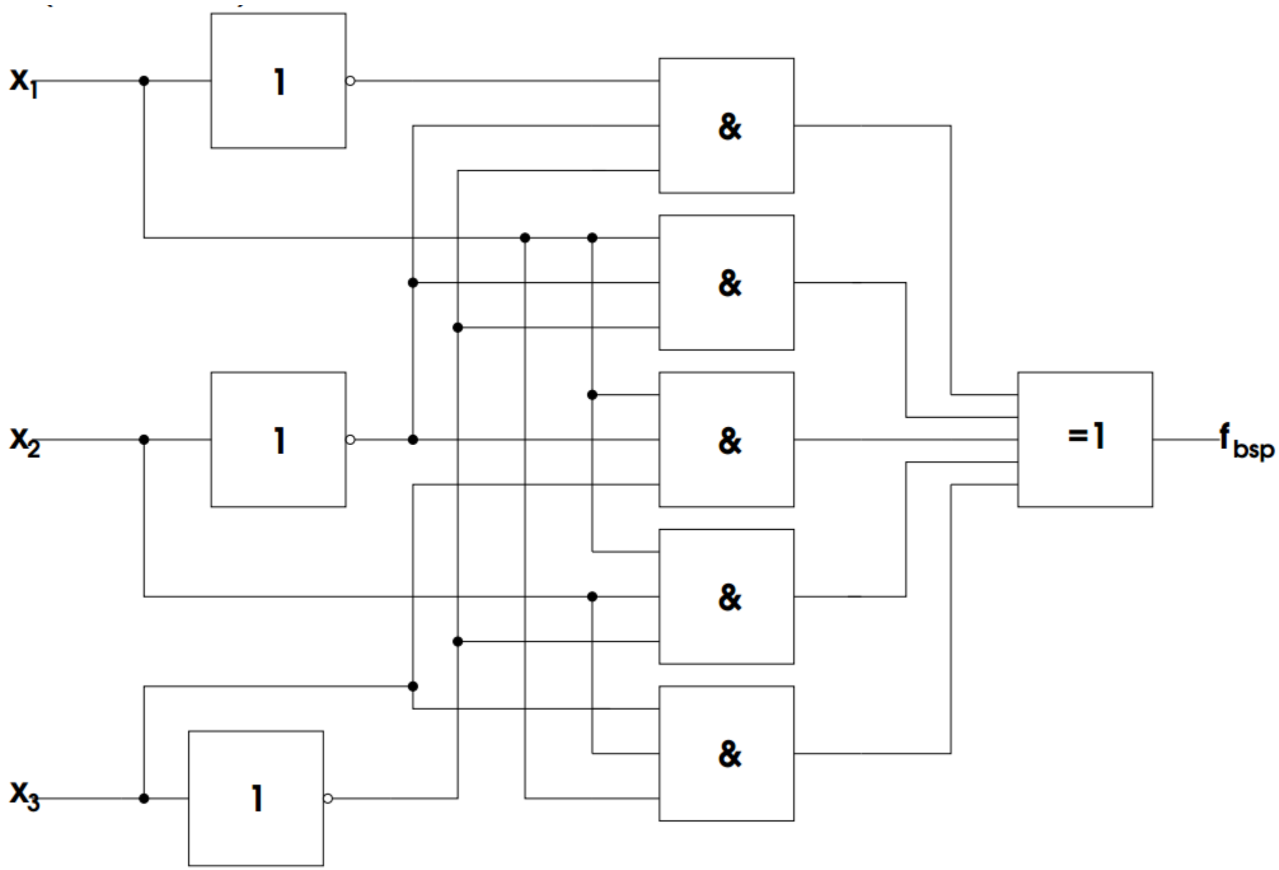


4.6 Schaltnetze

Beispiel RNF

$f_{bsp}: B^3 \rightarrow B,$ Wertevektor (1, 0, 0, 0, 1, 1, 1, 1)

$f_{bsp} = \overline{x}_1\overline{x}_2\,\overline{x}_3 \oplus x_1\,\overline{x}_2\,\overline{x}_3 \oplus x_1\,\overline{x}_2\,x_3 \oplus x_1\,x_2\,\overline{x}_3 \oplus x_1x_2\,x_3$

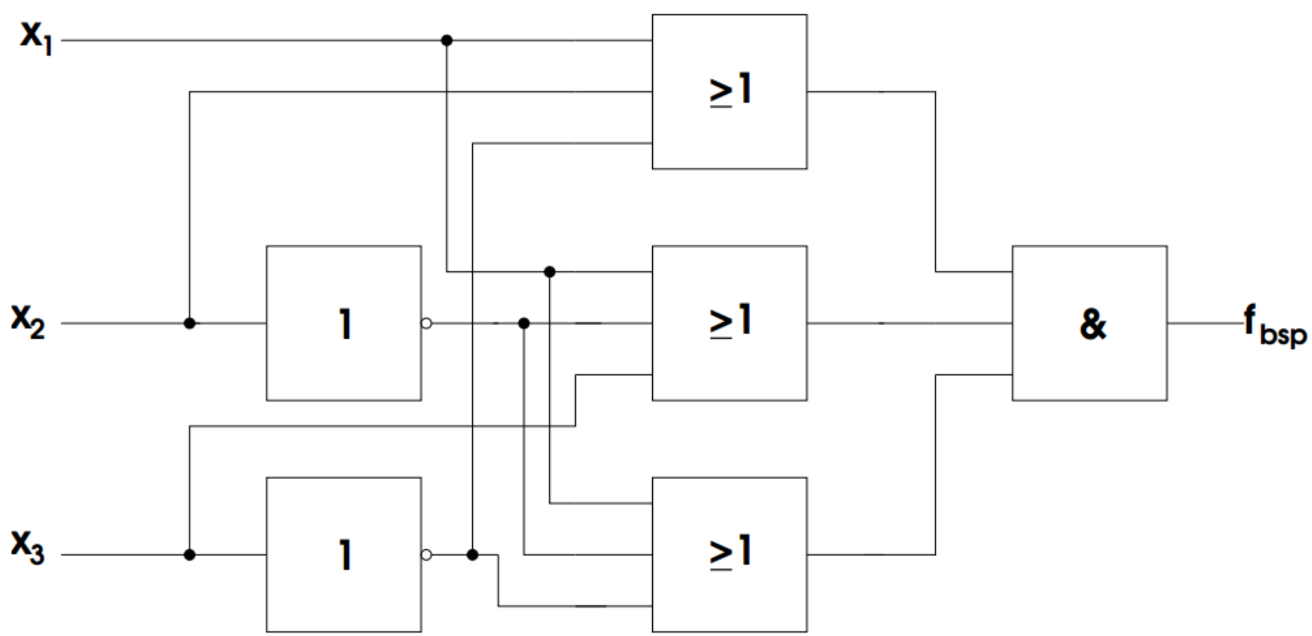


4.6 Schaltnetze

Beispiel KNF

$f_{bsp}: B^3 \rightarrow B,$ Wertevektor (1, 0, 0, 0, 1, 1, 1, 1)

$f_{bsp} = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3)$



4.6 Schaltnetze

Schaltnetzbewertung

Wir können nun beliebige Schaltnetze entwerfen. Wie messen wir die Qualität eines Schaltnetzes?

- **Schaltnetzgröße** (= Anzahl der Gatter) wegen **Kosten**, **Stromverbrauch**, **Verlustleistung**, **Zuverlässigkeit**, . . .
- **Schaltnetztiefe** (= Länge des längsten Wegs von Eingang zu Ausgang) wegen **Schaltgeschwindigkeit**
- **Fan-In** (= max. Anzahl eingehender Kanten) wegen **Realisierungsaufwand**
- **Fan-Out** (= max. Anzahl ausgehender Kanten) wegen **Realisierungsaufwand**
- . . . (z. B. Anzahl **Gattertypen**, **Testbarkeit**, **Verifizierbarkeit**)

4.6 Schaltnetze

Schaltnetzbewertung

Was wir schon wissen:

- Jede boolesche Funktion kann mit einem $\{\wedge, \vee, \neg\}$- bzw. einem $\{\oplus, \wedge, \neg\}$-Schaltnetz der Tiefe 3 realisiert werden

Beweis DNF, KNF oder RNF direkt umsetzen $\square$

Probleme

- Fan-In des tiefsten Gatters kann extrem groß sein
- Größe des Schaltnetzes oft inakzeptabel

4. Boolesche Funktionen und Schaltnetze

1. Einleitung ✓
2. Boolesche Algebra ✓
3. Repräsentationen boolescher Funktionen ✓
4. Normalformen boolescher Funktionen ✓
5. Repräsentation boolescher Funktionen mit OBDDs ✓
6. Schaltnetze ✓

Backup

4.6 Schaltnetze

Beispiel Multiplexer

$$MUX_d : B^{d+2^d} \rightarrow B$$
$$MUX_d\left(y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}\right) = x_{(y_1, y_2, \ldots, y_d)_2}$$

Wichtige Funktion für die Praxis

- Selektiert aus vielen Eingängen einen speziellen (ähnlich Drehschalter)
- kann parallel anliegende Daten in serielle Daten verwandeln
- mehrere Eingänge
  - Signaleingänge
  - Selektionseingänge
- ein Ausgang

4.6 Schaltnetze

Beispiel Multiplexer

$$MUX_d : B^{d+2^d} \to B$$
$$MUX_d(y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}) = x_{(y_1, y_2, \ldots, y_d)_2}$$

MUX_1 vollständige Darstellung

y_1	x_0	x_1	$MUX_1(y_1, x_0, x_1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

MUX_1 verkürzte Darstellung

y_1	$MUX_1(y_1, x_0, x_1)$
0	x_0
1	x_1

4.6 Schaltnetze

Beispiel Multiplexer

$$MUX_d: B^{d+2^d} \rightarrow B$$
$$MUX_d(y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}) = x_{(y_1, y_2, \ldots, y_d)_2}$$

MUX_3 verkürzte Darstellung

y_1	y_2	y_3	$MUX_3(y_1, y_2, y_3, x_0, x_1, \ldots, x_7)$
0	0	0	x_0
0	0	1	x_1
0	1	0	x_2
0	1	1	x_3
1	0	0	x_4
1	0	1	x_5
1	1	0	x_6
1	1	1	x_7

Beispiel Multiplexer

MUX_3 vollständige Darstellung

- $MUX_d : B^{d+2^d} \rightarrow B$
- $MUX_d\left(y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}\right) = x_{(y_1, y_2, \ldots, y_d)_2}$

- $MUX_3(y_1, y_2, y_3, x_0, x_1, \ldots, x_7) = x_{(y_1, y_2, y_3)_2}$
- $MUX_3 : B^{3+2^3} \rightarrow B$
- $MUX_3 : B^{11} \rightarrow B$

Die Abbildung $MUX_3 : B^{11} \rightarrow B$ hat demnach in vollständiger Darstellung 2^{11} = 2048 Zeilen

4.6 Schaltnetze

Beispiel Multiplexer

MUX_3 vollständige Darstellung

- $MUX_d: B^{d+2^d} \to B$
- $MUX_d\left(y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}\right) = x_{(y_1, y_2, \ldots, y_d)_2}$

- $MUX_3(y_1, y_2, y_3, x_0, x_1, \ldots, x_7) = x_{(y_1, y_2, y_3)_2}$
- $MUX_3: B^{3+2^3} \to B$
- $MUX_3: B^{11} \to B$

Die Abbildung $MUX_3: B^{11} \to B$ hat demnach in vollständiger Darstellung 2^{11} = 2048 Zeilen

Da die Hälfte der Indizes einschlägig ist (bedingt durch die Funktion eines Multiplexers) sind sowohl DNF als auch KNF riesig und würden zu sehr großen Schaltungen führen.

4.6 Schaltnetze

Beispiel Multiplexer

Trotzdem existiert eine überschaubare Schaltung für MUX_3.

- y_i und $\overline{y_i}$ selektieren passende UND-Gatter
- für jede Belegung der y_i ist genau ein UND-Gatter ausgewählt
- bis auf einen Eingang sind bei ausgewähltem Gatter alle Eingänge auf 1 gesetzt.
- Eigenschaft des Neutral-elements 1 für UND leitet sel. x_j durch