

05 – Boolesche Algebra & Boolesche Funktionen

Grundzüge digitaler Systeme

Vortrag von: Stefan Neumann

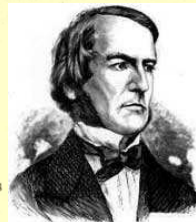
Boolesche Algebra & Boolesche Funktionen — Übersicht

- 1 Operationen der Booleschen Algebra
- 2 Gesetze der Booleschen Algebra
- 3 Funktionen über der Booleschen Algebra
- 4 Repräsentation boolescher Funktionen
 - If-Then-Else (ITE) Darstellung
 - Binäre Entscheidungsdiagramme / Binary Decision Diagrams (BDDs)

Boolesche Algebra

- Der englische Mathematiker George Boole (1815–1864) versuchte, Logik formal auszudrücken
 - Entwickelte dazu 1847 die Algebra der Logik: „Boolesche Algebra“
- Diese arbeitet mit den Werten falsche Aussage und wahre Aussage
- Abbildung auf 0 und 1

| | | |
|------------------------------------|--|--|
| The class X | x | |
| The class not-X | $1 - x$ | |
| All Xs are Ys All Ys are Xs | $x = y$ | |
| All Xs are Ys | $x(1 - y) = 0$ | |
| No Xs are Ys | $xy = 0$ | |
| All Ys are Xs Some Xs are Ys | $y = vx$ | $vx = \text{some Xs}$ $v(1 - x) = 0.$ |
| No Ys are Xs Some not-Xs are Ys | $y = v(1 - x)$ | $v(1 - x) = \text{some not-Xs}$ $vx = 0.$ |
| Some Xs are Ys | $\begin{cases} v = xy \\ \text{or } vx = vy \\ \text{or } vx(1 - y) = 0 \end{cases}$ | $v = \text{some Xs or some Ys}$ $vx = \text{some Xs, } vy = \text{some Ys}$ $v(1 - x) = 0, v(1 - y) = 0.$ |
| Some Xs are not Ys | $\begin{cases} v = x(1 - y) \\ \text{or } vx = v(1 - y) \\ \text{or } vxy = 0 \end{cases}$ | $v = \text{some Xs, or some not-Ys}$ $vx = \text{some Xs, } v(1 - y) = \text{some not-Ys}$ $v(1 - x) = 0, vy = 0.$ |



Operationen der Booleschen Algebra

| Notation | andere Schreibweise | Bezeichnung |
|----------|------------------------------|-------------|
| \wedge | $\cdot, \&$ | AND, UND |
| \vee | $+$ | OR, ODER |
| \neg | Überstreichung (\bar{e}) | NOT, NICHT |

$\left. \begin{array}{l} \text{AND, UND} \\ \text{OR, ODER} \end{array} \right\} \text{Binäre Operation (2 Operanden)}$
 \neg *Unäre Operation (1 Operand)*

| \wedge | 0 | 1 |
|----------|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| \vee | 0 | 1 |
|--------|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| \neg | |
|--------|---|
| 0 | 1 |
| 1 | 0 |

| a | b | $a \wedge b$ | $a \vee b$ | $\neg a$ |
|-----|-----|--------------|------------|----------|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Operatorrangfolge: \neg vor \wedge vor \vee

Boolesche Algebra & Boolesche Funktionen — Übersicht

- 1 Operationen der Booleschen Algebra
- 2 Gesetze der Booleschen Algebra
- 3 Funktionen über der Booleschen Algebra
- 4 Repräsentation boolescher Funktionen
 - If-Then-Else (ITE) Darstellung
 - Binäre Entscheidungsdiagramme / Binary Decision Diagrams (BDDs)

Gesetze der Booleschen Algebra

■ Kommutativgesetz

- $x \vee y = y \vee x$

- $x \wedge y = y \wedge x$

■ Assoziativgesetz

- $(x \vee y) \vee z = x \vee (y \vee z)$

- $(x \wedge y) \wedge z = x \wedge (y \wedge z)$

■ Distributivgesetz

- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

- $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

■ Absorptionsgesetz

- $x \vee (x \wedge y) = x$

- $x \wedge (x \vee y) = x$

■ Idempotenz

- $(x \wedge x) = x$

- $(x \vee x) = x$

■ Doppelnegation

- $\neg\neg x = x$

■ Weitere Gesetze

- $0 \vee x = x$ bzw. $1 \wedge x = x$

- $x \vee \neg x = 1$ bzw. $x \wedge \neg x = 0$

Gesetze der Booleschen Algebra

■ **Beweis:** Aus $(x \wedge y = 0)$ und $(x \vee y = 1)$ folgt $(y = \neg x)$

■ Dies kann folgendermaßen gezeigt werden:

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

■ $(y \vee x) = 1$ setzen:

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

■ Außerdem:

$$\begin{aligned}\neg x &= \neg x \vee 0 \\&= \neg x \vee (x \wedge y) \\&= (\neg x \vee x) \wedge (\neg x \vee y) \\&= 1 \wedge (\neg x \vee y) \\&= \neg x \vee y = y \vee \neg x\end{aligned}$$

■ Woraus die Gleichheit $(y = \neg x)$ folgt

Gesetze der Booleschen Algebra

De Morgan

- Die de Morganschen Gesetze:

- $\neg(x \vee y) = \neg x \wedge \neg y$

- $\neg(x \wedge y) = \neg x \vee \neg y$

- Man kann die de Morganschen Gesetze mittels Wahrheitstabelle beweisen:

| x | y | $x \wedge y$ | $\neg(x \wedge y)$ | $\neg x$ | $\neg y$ | $\neg x \vee \neg y$ |
|---|---|--------------|--------------------|----------|----------|----------------------|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Gesetze der Booleschen Algebra

- Erweiterung des Gesetzes von de Morgan auf n Variablen:

$$x_1 \vee x_2 \vee x_3 \vee \dots \vee x_n = \bigvee_{i=1}^n x_i$$

bzw.

$$x_1 \wedge x_2 \wedge x_3 \wedge \dots \wedge x_n = \bigwedge_{i=1}^n x_i$$

- Durch vollständige Induktion nach n erhalten wir dann:

$$\neg \bigvee_{i=1}^n x_i = \bigwedge_{i=1}^n \neg x_i$$

bzw.

$$\neg \bigwedge_{i=1}^n x_i = \bigvee_{i=1}^n \neg x_i$$

Boolesche Algebra & Boolesche Funktionen — Übersicht

- 1 Operationen der Booleschen Algebra
- 2 Gesetze der Booleschen Algebra
- 3 Funktionen über der Booleschen Algebra
- 4 Repräsentation boolescher Funktionen
 - If-Then-Else (ITE) Darstellung
 - Binäre Entscheidungsdiagramme / Binary Decision Diagrams (BDDs)

Funktionen über der Booleschen Algebra

- Sei $e_i \in \{0, 1\}$ für $i = 1, 2, \dots, n$ und sei $a \in \{0, 1\}$
- Betrachte Funktion

$$f: (e_1, e_2, \dots, e_n) \rightarrow a$$

- Für $n = 1$: Eine Eingangsvariable e wird auf die Ausgangsvariable a_* abgebildet
 - Tabelle definiert vier verschiedene Funktionen $f_{1,j}: (e) \rightarrow (a_{1,j})$ für $j = 0, \dots, 4$
 - Logische Funktionen mit einer Eingangsvariablen und einer Ausgangsvariablen
 - Beispiel zum Lesen der Tabelle: $f_{1,2}$ erfüllt $f_{1,2}(0) = 1$ und $f_{1,2}(1) = 0$

| | | $e:$ | | | |
|-----|-----------|------|---|------------------------|--------------|
| | | 0 | 1 | | |
| j | | | | $a_{1,j} = f_{1,j}(e)$ | Funktionstyp |
| 0 | $f_{1,0}$ | 0 | 0 | $a_{1,0} = 0$ | Nullfunktion |
| 1 | $f_{1,1}$ | 0 | 1 | $a_{1,1} = e$ | Identität |
| 2 | $f_{1,2}$ | 1 | 0 | $a_{1,2} = \neg e$ | Negation |
| 3 | $f_{1,3}$ | 1 | 1 | $a_{1,3} = 1$ | Einsfunktion |

Funktionen über der Booleschen Algebra

Funktionen mit zwei Eingangsvariablen

- Erweiterung der Tabelle auf mehr als eine Variable:

- Die Funktion $f_{1.6}$ (XOR) erfüllt:

- $f_{1.6}(0, 0) = 0$
- $f_{1.6}(1, 0) = 1$
- $f_{1.6}(0, 1) = 1$
- $f_{1.6}(1, 1) = 0$

| | | | | | | | | |
|-----|-----------|---------|---|---|---|---|---|---|
| | | $e_1 :$ | 0 | 1 | 0 | 1 | | |
| | | $e_2 :$ | 0 | 0 | 1 | 1 | | |
| j | | | | | | | $a_{2,j} = f_{2,j}(e_1, e_2)$ | Name |
| 6 | $f_{2.6}$ | 0 | 1 | 1 | 0 | | $a_{2.6} = (e_1 \vee e_2) \wedge \neg(e_1 \wedge e_2) = e_1 \not\equiv e_2$ | Antivalenz, XOR, \oplus , $\not\equiv$, \nrightarrow , \nleftarrow |

- Anhand solcher Tabellen kann man sehen (siehe nächster Slide), dass es gibt nur 16 verschiedene Funktionen über zwei Variablen gibt

Funktionen über der Booleschen Algebra

Funktionen mit zwei Eingangsvariablen

| | $e_1 :$ | 0 | 1 | 0 | 1 | | |
|-----|------------|---|---|---|---|--|---|
| | $e_2 :$ | 0 | 0 | 1 | 1 | | |
| j | | | | | | $a_{2,j} = f_{2,j}(e_1, e_2)$ | Name |
| 0 | $f_{2,0}$ | 0 | 0 | 0 | 0 | $a_{2,0} = 0$ | Nullfunktion, \perp |
| 1 | $f_{2,1}$ | 0 | 0 | 0 | 1 | $a_{2,1} = e_1 \wedge e_2$ | Konjunktion, AND, \wedge |
| 2 | $f_{2,2}$ | 0 | 0 | 1 | 0 | $a_{2,2} = \neg e_1 \wedge e_2 = e_1 \nrightarrow e_2$ | Inhibition von e_2 |
| 3 | $f_{2,3}$ | 0 | 0 | 1 | 1 | $a_{2,3} = e_2$ | Identität von e_2 |
| 4 | $f_{2,4}$ | 0 | 1 | 0 | 0 | $a_{2,4} = e_1 \wedge \neg e_2 = e_1 \not\leftarrow e_2$ | Inhibition von e_1 |
| 5 | $f_{2,5}$ | 0 | 1 | 0 | 1 | $a_{2,5} = e_1$ | Identität von e_1 |
| 6 | $f_{2,6}$ | 0 | 1 | 1 | 0 | $a_{2,6} = (e_1 \vee e_2) \wedge \neg(e_1 \wedge e_2) = e_1 \neq e_2$ | Antivalenz, XOR, \oplus , \neq , \nrightarrow , \nleftarrow |
| 7 | $f_{2,7}$ | 0 | 1 | 1 | 1 | $a_{2,7} = e_1 \vee e_2$ | Disjunktion, OR, \vee |
| 8 | $f_{2,8}$ | 1 | 0 | 0 | 0 | $a_{2,8} = \neg e_1 \wedge \neg e_2 = \neg(e_1 \vee e_2) = e_1 \downarrow e_2$ | NOR, \downarrow |
| 9 | $f_{2,9}$ | 1 | 0 | 0 | 1 | $a_{2,9} = (e_1 \wedge e_2) \vee \neg(e_1 \vee e_2) = e_1 \Leftrightarrow e_2$ | Äquivalenz, \equiv , \leftrightarrow , \Leftrightarrow |
| 10 | $f_{2,10}$ | 1 | 0 | 1 | 0 | $a_{2,10} = \neg e_1$ | Negation von e_1 |
| 11 | $f_{2,11}$ | 1 | 0 | 1 | 1 | $a_{2,11} = \neg e_1 \vee e_2 = e_1 \Rightarrow e_2$ | Implikation, \supset , \rightarrow , \Rightarrow |
| 12 | $f_{2,12}$ | 1 | 1 | 0 | 0 | $a_{2,12} = \neg e_2$ | Negation von e_2 |
| 13 | $f_{2,13}$ | 1 | 1 | 0 | 1 | $a_{2,13} = e_1 \vee \neg e_2 = e_1 \Leftarrow e_2$ | Replikation, \subset , \leftarrow , \Leftarrow |
| 14 | $f_{2,14}$ | 1 | 1 | 1 | 0 | $a_{2,14} = \neg e_1 \vee \neg e_2 = \neg(e_1 \wedge e_2) = e_1 \uparrow e_2$ | NAND, \uparrow |
| 15 | $f_{2,15}$ | 1 | 1 | 1 | 1 | $a_{2,15} = 1$ | Einsfunktion, \top |

Funktionen über der Booleschen Algebra

Implikation (oder Subjunktion)

- Symbol: „ \Rightarrow “ bzw. „ \rightarrow “ bzw. „ \supset “
- $(e_1 \Rightarrow e_2)$ entspricht dem Ausdruck $(\neg e_1 \vee e_2)$

| e_1 | e_2 | $e_1 \Rightarrow e_2$ |
|-------|-------|-----------------------|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Funktionen über der Booleschen Algebra

Äquivalenz (oder Bijunktion)

- Symbol: „ \Leftrightarrow “ bzw. „ \leftrightarrow “ bzw. „ \equiv “
- $(e_1 \Leftrightarrow e_2)$ entspricht dem Ausdruck $(e_1 \Rightarrow e_2) \wedge (e_2 \Rightarrow e_1)$

| | | $\neg e_1 \vee e_2$ | $\neg e_2 \vee e_1$ | |
|-------|-------|-----------------------|-----------------------|---------------------------|
| e_1 | e_2 | $e_1 \Rightarrow e_2$ | $e_2 \Rightarrow e_1$ | $e_1 \Leftrightarrow e_2$ |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Funktionen über der Booleschen Algebra

NAND und NOR

- NAND (NOT AND, verneintes UND)
 - $(e_1 \text{ NAND } e_2)$ entspricht dem Ausdruck $\neg(e_1 \wedge e_2)$
- NOR (NOT OR, verneintes ODER)
 - $(e_1 \text{ NOR } e_2)$ entspricht dem Ausdruck $\neg(e_1 \vee e_2)$

| e_1 | e_2 | $e_1 \text{ NAND } e_2$ | $e_1 \text{ NOR } e_2$ |
|-------|-------|-------------------------|------------------------|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

Funktionen über der Booleschen Algebra

Tautologie

- *Tautologie*: Formel, die immer 1 liefert (egal, wie Variablen belegt sind)
 - Gültige Formel
- Beispiele: $(e_1 \vee \neg e_1)$, $(e \Rightarrow e)$, $(e_1 \wedge e_2) \Rightarrow e_1$

Aufstellen der *Wahrheitstabellen*:

| e | $e \vee \neg e$ | $e \Rightarrow e$ |
|-----|-----------------|-------------------|
| 0 | 1 | 1 |
| 1 | 1 | 1 |

| e_1 | e_2 | $e_1 \wedge e_2$ | $(e_1 \wedge e_2) \Rightarrow e_1$ |
|-------|-------|------------------|------------------------------------|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Funktionen über der Booleschen Algebra

Kontradiktion

- *Kontradiktion*: Formel, die immer 0 liefert (egal, wie Variablen belegt sind)
 - Unerfüllbare Formel
- Beispiele: $(e \wedge \neg e)$, $(e \Leftrightarrow \neg e)$

| e | $\neg e$ | $e \wedge \neg e$ | $e \Leftrightarrow \neg e$ |
|-----|----------|-------------------|----------------------------|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |

Funktionen über der Booleschen Algebra

Funktionale Vollständigkeit

Definition Funktionale Vollständigkeit

Eine Teilmenge der Booleschen Funktionen ist *funktional vollständig*, wenn alle Booleschen Funktionen durch Kombinationen dieser Funktionen ausgedrückt werden können.

Beispiele für funktional vollständige Mengen:

- Die Menge $\{\neg, \wedge, \vee\}$ ist funktional vollständig.

- z.B.: $(x \Rightarrow y) = (\neg x \vee y)$

- Die Menge $\{\neg, \wedge\}$ ist funktional vollständig.

Funktionen über der Booleschen Algebra

Funktionale Vollständigkeit

Satz: Die Menge $\{\neg, \wedge\}$ ist funktional vollständig.

Beweis:

- Wir wissen, dass die Menge $\{\neg, \wedge, \vee\}$ vollständig ist
- Es genügt also zu zeigen, dass \vee mit Hilfe von \neg und \wedge ausgedrückt werden kann
- Behauptung: Es gilt $x \vee y = \neg(\neg x \wedge \neg y)$
- Das können wir mit einer Wahrheitstafel überprüfen:

| x | y | $\neg x$ | $\neg y$ | $\neg x \wedge \neg y$ | $\neg(\neg x \wedge \neg y)$ |
|-----|-----|----------|----------|------------------------|------------------------------|
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |

Funktionen über der Booleschen Algebra

Funktionale Vollständigkeit

- **Satz:** Die Menge $\{\text{NAND}\}$ ist funktional vollständig.
- **Beweis:**
 - $\{\neg, \wedge\}$ ist funktional vollständig (siehe vorheriger Slide)
 - $\neg x = x \text{ NAND } x$
 - $x \wedge y = \neg(\neg(x \wedge y)) = \neg(x \text{ NAND } y) = (x \text{ NAND } y) \text{ NAND } (x \text{ NAND } y)$
- Praktisch relevant: NAND ist einfach als Halbleiter-Schaltkreis realisierbar
 - ⇒ Somit ist jede logische Funktion als Schaltkreis realisierbar
- Die Mengen $\{\text{NOR}\}$, $\{\neg, \vee\}$, $\{\neg, \Rightarrow\}$ und $\{\Rightarrow, \perp\}$ sind ebenfalls funktional vollständig

Normalformen

Normalform

Eine **Normalform** ist

- eine Einschränkung auf der Syntax, sodass
 - jede beliebige Formel in eine semantisch äquivalente Formel in Normalform umgewandelt werden kann.
-
- Vereinfacht die (maschinelle) Verarbeitung von logischen Formeln
 - Viele Verfahren/Tools setzen voraus, dass Formeln in einer bestimmten Normalform sind

Definition

Eine Formel ist in **Negationsnormalform (NNF)**, wenn nur \wedge , \vee , \neg verwendet werden und Negationen nur direkt vor Variablen stehen.

Beispiel: $e_1 \vee (\neg e_2 \wedge (\neg e_1 \vee e_3))$

Normalformen

Konjunktive Normalform

Literal: Unter Literalen verstehen wir

- Variablen x und
- negierte Variablen $\neg x$

Definition

Eine Formel ist in **konjunktiver Normalform (KNF)**, wenn sie eine Konjunktion von Disjunktionen von Literalen ist. F ist also von folgender Form ($L_{i,j}$ Literale):

$$F = (L_{1,1} \vee \cdots \vee L_{1,m_1}) \wedge \cdots \wedge (L_{n,1} \vee \cdots \vee L_{n,m_n})$$

Beispiele:

- $(\neg a \vee b) \wedge (a \vee \neg c \vee d) \wedge \neg b$
- $(\neg a \vee x \vee \neg c \vee d) \wedge (\neg b \vee \neg y)$

Normalformen

Disjunktive Normalform

Definition

Eine Formel ist in **disjunktiver Normalform (DNF)**, wenn sie eine Disjunktion von Konjunktionen von Literalen ist. F ist also von folgender Form ($L_{i,j}$ Literale):

$$F = (L_{1,1} \wedge \cdots \wedge L_{1,m_1}) \vee \cdots \vee (L_{n,1} \wedge \cdots \wedge L_{n,m_n})$$

Beispiele:

- $(\neg a \vee b) \wedge (a \vee \neg c \vee d) \wedge \neg b$ (KNF)
- $(\neg a \wedge b \wedge c) \vee (a \wedge \neg c \wedge d)$ (DNF)
- $\neg a \wedge b \wedge c$ (DNF, KNF)
- $\neg a \vee b \vee c$ (DNF, KNF)
- $(\neg a \vee b \vee c) \wedge (a \vee (b \wedge d))$ ()

Jede Formel kann in eine semantisch äquivalente KNF (bzw. DNF) transformiert werden.

Normalformen

kanonische KNF/ DNF

- *Volldisjunktion bzw. Maxterm*: Disjunktion von Literalen über alle Variablen
- *Vollkonjunktion bzw. Minterm*: Konjunktion von Literalen über alle Variablen

Definition

Eine Formel ist in **kanonischer konjunktiver Normalform (KKNF)**, wenn sie eine Konjunktion von paarweise verschiedenen Volldisjunktionen ist.

$$\text{Bsp: } f(e_1, e_2, e_3, e_4) = (e_1 \vee \neg e_2 \vee e_3 \vee \neg e_4) \wedge (\neg e_1 \vee \neg e_2 \vee e_3 \vee \neg e_4)$$

Definition

Eine Formel ist in **kanonischer disjunktiver Normalform (KDNF)**, wenn sie eine Disjunktion von paarweise verschiedenen Vollkonjunktionen ist.

$$\text{Bsp: } f(e_1, e_2, e_3, e_4) = (e_1 \wedge \neg e_2 \wedge e_3 \wedge \neg e_4) \vee (\neg e_1 \wedge \neg e_2 \wedge e_3 \wedge \neg e_4)$$

Normalformen

Beispiel: von der Wahrheitstabelle zu Normalformen

| e_1 | e_2 | e_3 | $f(e_1, e_2, e_3)$ | Maxterm / Minterm |
|-------|-------|-------|--------------------|---------------------------------------|
| 0 | 0 | 0 | 0 | $e_1 \vee e_2 \vee e_3$ |
| 0 | 0 | 1 | 0 | $e_1 \vee e_2 \vee \neg e_3$ |
| 0 | 1 | 0 | 1 | $\neg e_1 \wedge e_2 \wedge \neg e_3$ |
| 0 | 1 | 1 | 1 | $\neg e_1 \wedge e_2 \wedge e_3$ |
| 1 | 0 | 0 | 0 | $\neg e_1 \vee e_2 \vee e_3$ |
| 1 | 0 | 1 | 1 | $e_1 \wedge \neg e_2 \wedge e_3$ |
| 1 | 1 | 0 | 1 | $e_1 \wedge e_2 \wedge \neg e_3$ |
| 1 | 1 | 1 | 1 | $e_1 \wedge e_2 \wedge e_3$ |

$$f(e_1, e_2, e_3) = (e_1 \vee e_2 \vee e_3) \wedge (e_1 \vee e_2 \vee \neg e_3) \wedge (\neg e_1 \vee e_2 \vee e_3) \quad (\text{KKNF})$$

$$= (e_1 \vee e_2) \wedge (e_2 \vee e_3) \quad (\text{min. KNF})$$

$$= (\neg e_1 \wedge e_2 \wedge \neg e_3) \vee (\neg e_1 \wedge e_2 \wedge e_3) \vee (e_1 \wedge \neg e_2 \wedge e_3) \vee (e_1 \wedge e_2 \wedge \neg e_3) \vee (e_1 \wedge e_2 \wedge e_3) \quad (\text{KDNF})$$

$$= (e_1 \wedge e_3) \vee e_2 \quad (\text{min. DNF})$$

Normalformen

Konjunktive und Disjunktive Normalform - Beispiele

Zwei Beispiele wie man eine Formel in KNF umformen kann:

$$(A \wedge B) \Rightarrow (C)$$

$$(\neg(A \wedge B) \vee C) \quad (\text{Definition von } \Rightarrow)$$

$$((\neg A \vee \neg B) \vee C) \quad (\text{de Morgan})$$

$$(\neg A \vee \neg B \vee C) \quad (\text{Assoziativität})$$

$$C \Leftrightarrow (A \wedge B)$$

$$(C \Rightarrow (A \wedge B)) \wedge ((A \wedge B) \Rightarrow C) \quad (\text{Definition von } \Leftrightarrow)$$

$$(\neg C \vee (A \wedge B)) \wedge (C \vee \neg(A \wedge B)) \quad (\text{Definition von } \Rightarrow)$$

$$(\neg C \vee (A \wedge B)) \wedge (C \vee (\neg A \vee \neg B)) \quad (\text{de Morgan})$$

$$(\neg C \vee (A \wedge B)) \wedge (C \vee \neg A \vee \neg B) \quad (\text{Assoziativität})$$

$$((\neg C \vee A) \wedge (\neg C \vee B)) \wedge (C \vee \neg A \vee \neg B) \quad (\text{Distributivität})$$

$$(\neg C \vee A) \wedge (\neg C \vee B) \wedge (C \vee \neg A \vee \neg B) \quad (\text{Assoziativität})$$

Normalformen

Konjunktive und Disjunktive Normalform - Algorithmus

Diese Schritte lassen sich auch in einen Algorithmus fassen der eine Formel F in eine semantisch äquivalente KNF umwandelt.

Algorithmus

- 1 Ersetze alle Junktoren durch \wedge , \vee und \neg .
 - Ersetze $G \Leftrightarrow H$ durch $(G \Rightarrow H) \wedge (H \Rightarrow G)$
 - Ersetze $G \Rightarrow H$ durch $(\neg G \vee H)$
 - ...
- 2 Iteriere das Folgende solange wie möglich
 - Ersetze $\neg\neg G$ durch G
 - Ersetze $\neg(G \wedge H)$ durch $(\neg G \vee \neg H)$
 - Ersetze $\neg(G \vee H)$ durch $(\neg G \wedge \neg H)$
- 3 Iteriere das Folgende solange wie möglich
 - Ersetze $(G \vee (H \wedge I))$ und $((H \wedge I) \vee G)$ durch $((G \vee H) \wedge (G \vee I))$

„Ersetze“ liest sich als „Ersetze alle Teilformeln der Form“

Boolesche Algebra & Boolesche Funktionen — Übersicht

- 1 Operationen der Booleschen Algebra
- 2 Gesetze der Booleschen Algebra
- 3 Funktionen über der Booleschen Algebra
- 4 **Repräsentation boolescher Funktionen**
 - If-Then-Else (ITE) Darstellung
 - Binäre Entscheidungsdiagramme / Binary Decision Diagrams (BDDs)

Repräsentation Boolescher Funktionen

- Wie kann man Boolesche Funktionen repräsentieren und prüfen, ob zwei Formeln äquivalent sind?
- Wahrheitstabelle
 - “Semantische Methode”
 - Siehe zum Beispiel Slides 17 und 19
 - Nachteil: Für n Variablen haben wir 2^n Zeilen in der Wahrheitstabelle
 - Wir haben *immer* 2^n viele Zeilen
 - Nur anwendbar, wenn n klein ist
- Textuelle Darstellung
 - “Algebraische Methode”
 - Darstellung der Formel und Vereinfachung mittels algebraischer Regeln
 - Beispielweise Umwandlung in Normalform mittels Algorithmus von Slide 28
 - Nachteil: Vereinfachung umständlich
 - Jede Anwendung des Distributivgesetz kann Formellänge potentiell verdoppeln
 - Im schlimmsten Fall exponentiell (in n) lange Formeln
 - In manchen Fällen funktioniert Vereinfachung aber gut

Repräsentation Boolescher Funktionen

- Tools zur Vereinfachung
 - Bauen auf konjunktiver oder disjunktiver Normalform auf
 - Für manche Boolesche Funktionen suboptimal
- Eigenschaften anderer Darstellungen:
 - Besser geeignet für Vereinfachungen
 - KV-Diagramme (Karnaugh-Veitch-Diagramme, Karnaugh map)
 - Geeignet zur Implementierung in Software (Hochsprache, Maschinencode)
 - **If-Then-Else (ITE)**
 - Kompakte Darstellung, bei der Operationen direkt ausgeführt werden
 - **Binäre Entscheidungsdiagramme** (Binary Decision Diagrams, BDDs)

ITE-Darstellung Boolescher Funktionen

- Darstellung mittels If-Then-Else (ITE)
- Zum Beispiel: $(a \wedge \neg b) \vee \neg a$ dargestellt als

```
if (a == true) then
  if (b == true) then
    false
  else
    true
else
  true
```

- Geeignet zur Implementierung in Software (Hochsprache, Maschinencode)
- Ableitbar aus Binary Decision Diagram

- Einführung des ITE Operators
 - $\text{ITE}(a, b, c)$ kann interpretiert werden als `if (a == true) then b else c`
 - Darstellung Boolescher Operatoren
 - $a \equiv \text{ITE}(a, 1, 0)$
 - $a \wedge b \equiv \text{ITE}(a, b, 0)$
 - $\neg a \equiv \text{ITE}(a, 0, 1)$
 - $a \vee b \equiv \text{ITE}(a, 1, b)$
- Umwandlung von Disjunktiver-/Konjunktiver Normalform in ITE
 - Basiert auf Shannon-Zerlegung

Wahrheitstabelle und ITE-Form

| <i>a</i> | <i>b</i> | <i>c</i> | <i>y</i> |
|----------|----------|----------|----------|
| 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 |

```
if (a == true) then
  if (b == true) then
    if (c == true) then 1
    else 0
  else
    if (c == true) then 1
    else 0
else
  if (b == true) then
    if (c == true) then 1
    else 1
  else
    if (c == true) then 0
    else 0
```

Vereinfachung von Formeln: Motivation

- Betrachten wir folgende Formel:

$$\begin{aligned} f = & (a \vee b \vee c \vee d) \wedge (a \vee x \vee y \vee z) \\ & \wedge (a \vee \neg x \vee \neg w \vee z) \wedge (a \vee \neg x \vee \neg c \vee \neg z) \\ & \wedge (\neg b \vee x \vee c \vee \neg x) \wedge (\neg b \vee \neg y \vee e \vee \neg g) \\ & \wedge (\neg b \vee y \vee e \vee \neg g) \wedge (\neg b \vee y \vee e \vee g) \end{aligned}$$

- Auf den ersten Blick nicht klar, ob Formel erfüllbar ist oder nicht
- Trick: Variablenwerte “raten” und Werte einsetzen
- Beispiel: Wir setzen oben $a = 1$:

$$\begin{aligned} f = & 1 \wedge 1 \wedge 1 \wedge 1 \wedge (\neg b \vee x \vee c \vee \neg x) \wedge (\neg b \vee \neg y \vee e \vee \neg g) \\ & \wedge (\neg b \vee y \vee e \vee \neg g) \wedge (\neg b \vee y \vee e \vee g) \end{aligned}$$

- Beispiel: Wenn wir nun $b = 0$ raten, erhalten wir $f = 1$
- ⇒ Die Funktionen, die durch das “Einsetzen” von einzelnen Variablen entstehen, nennt man **Kofaktoren** (siehe nächster Slide)

Kofaktoren

- Sei $f: \{0, 1\}^n \rightarrow \{0, 1\}$ eine Boolesche Funktion und betrachte $f(x_1, \dots, x_n)$
- Definiere **Kofaktor $f_{x_i,1}$ von f nach $x_i = 1$** durch

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

⇒ $f_{x_i,1}$ ist die Funktion, die man erhält, wenn man $x_i = 1$ in f einsetzt

- Beispiel: Für $f = (a \vee b \vee c \vee d) \wedge (a \vee x \vee y \vee z) \wedge (\neg b \vee y \vee e \vee \neg g)$ gilt

$$\begin{aligned} f_{a,1} &= (1 \vee b \vee c \vee d) \wedge (1 \vee x \vee y \vee z) \wedge (\neg b \vee y \vee e \vee \neg g) \\ &= 1 \wedge 1 \wedge (\neg b \vee y \vee e \vee \neg g) = (\neg b \vee y \vee e \vee \neg g) \end{aligned}$$

- Entsprechend heißt $f_{x_i,0}$ **Kofaktor von f nach $x_i = 0$**

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

⇒ $f_{x_i,0}$ ist die Funktion, die man erhält, wenn man $x_i = 0$ in f einsetzt

- Beispiel: Für f wie oben gilt:

$$\begin{aligned} f_{b,0} &= (a \vee b \vee c \vee d) \wedge (a \vee x \vee y \vee z) \wedge (\neg 0 \vee y \vee e \vee \neg g) \\ &= (a \vee b \vee c \vee d) \wedge (a \vee x \vee y \vee z) \end{aligned}$$

Shannon-Zerlegung

- Für jede Formel f gilt die **Shannon-Zerlegung**: $f = (x_i \wedge f_{x_i,1}) \vee (\neg x_i \wedge f_{x_i,0})$
 - $x_i \wedge f_{x_i,1}$ entspricht dem Fall wenn $x_i = 1$
 - $\neg x_i \wedge f_{x_i,0}$ entspricht dem Fall wenn $x_i = 0$
 - Interpretation:
 - Wir fixieren $x_i = 1$ und verwenden Kofaktor $f_{x_i,1}$ *oder* wir fixieren $x_i = 0$ und verwenden Kofaktor $f_{x_i,0}$
 - Erlaubt das “systematische Explorieren” von Variablenwerten
 - Besonders hilfreich für Variablen, die häufig vorkommen
 - Ist das gleiche wie $\text{ITE}(x_i, f_{x_i,1}, f_{x_i,0})$

Shannon-Zerlegung - Beispiel

Berechne Shannon-Zerlegung von f :

$$\begin{aligned} f &= (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge b \wedge c) \\ &= (a \wedge f_{a,1}) \vee (\neg a \wedge f_{a,0}) \end{aligned}$$

$$g := f_{a,1} = (\neg b \wedge c) \vee (b \wedge c)$$

$$h := f_{a,0} = (b \wedge \neg c) \vee (b \wedge c)$$

Jetzt Shannon-Zerlegungen von $g = f_{a,1}$ und von $h = f_{a,0}$ nach b berechnen:

$$g = (b \wedge \underbrace{c}_{g_{b,1}}) \vee (\neg b \wedge \underbrace{c}_{g_{b,0}}) = \text{ITE}(b, c, c) = c$$

$$h = (b \wedge \underbrace{(\neg c \vee c)}_{h_{b,1}}) \vee (\neg b \wedge \underbrace{0}_{h_{b,0}}) = \text{ITE}(b, 1, 0) = b$$

Jetzt Darstellung mit Hilfe von ITE-Statements:

$$\begin{aligned} f &= \text{ITE}(a, f_{a,1}, f_{a,0}) \\ &= \text{ITE}(a, g, h) \\ &= \text{ITE}(a, c, b) \end{aligned}$$

Binäre Entscheidungsdiagramme / Binary Decision Diagrams (BDD)

- Binäre Entscheidungsdiagramme sind eine weitere Darstellung von Booleschen Funktionen
 - Englisch: Binary Decision Diagrams (BDD)
 - Boolesche Funktionen werden als Flussdiagramm dargestellt, das Auswertung der Funktion erlaubt
 - Führt teilweise zu schnellerer Auswertung von Funktionen
 - Kann auch als Form der Komprimierung der Booleschen Funktion angesehen werden

BDD – Beispiel

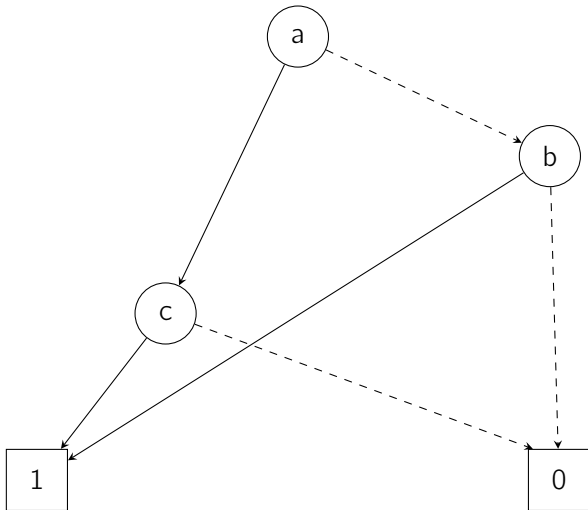
$$f = (a \wedge c) \vee (\neg a \wedge b)$$

Negation auf der Kante

—→ nicht negiert

- - - - -→ negiert

Pro „Ebene“ eine Variable
(hier zufällig: pro Ebene ein Knoten)



Grundsätzlich zwei Möglichkeiten BDDs zu erstellen:

- Ausgehend von Entscheidungsbaum (Decision Tree) + Vereinfachung
- Aus Wahrheitstabelle mit sog. Beads (basiert auf Shannon-Zerlegung)

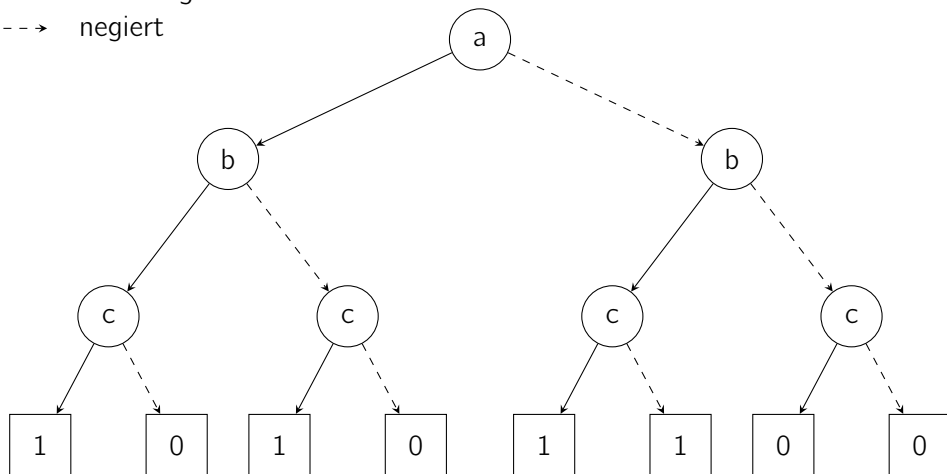
Decision Tree

Beispiel von Folie 34

Negation auf der Kante

————→ nicht negiert

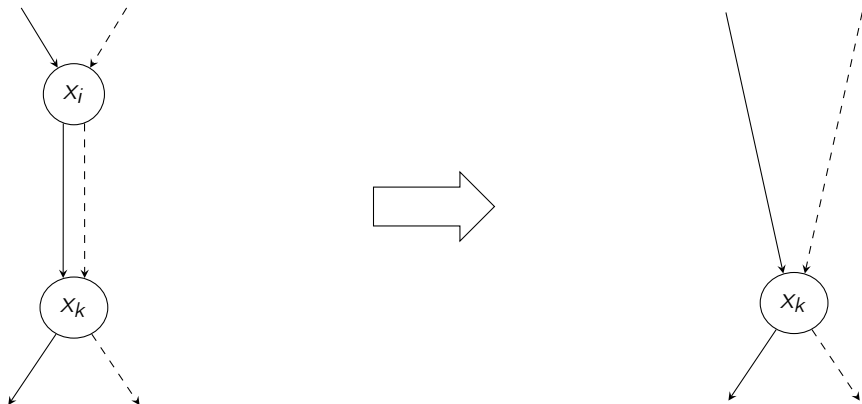
- - - - -> negiert



- Umwandlung eines Decision Trees in einen minimalen DAG (Directed Acyclic Graph)
- Nur ein 0 und ein 1 Knoten, danach wiederholtes Anwenden von zwei Regeln bis nicht mehr möglich:
 - **Delete Rule:** Löschen unnötiger Zwischenstufen
 - **Merge Rule:** Zusammenfassen von identen Teilgraphen
- Ergebnis: Binary Decision Diagram (BDD)
 - Eigentlich geordnetes BDD, engl. Ordered BDD = OBDD

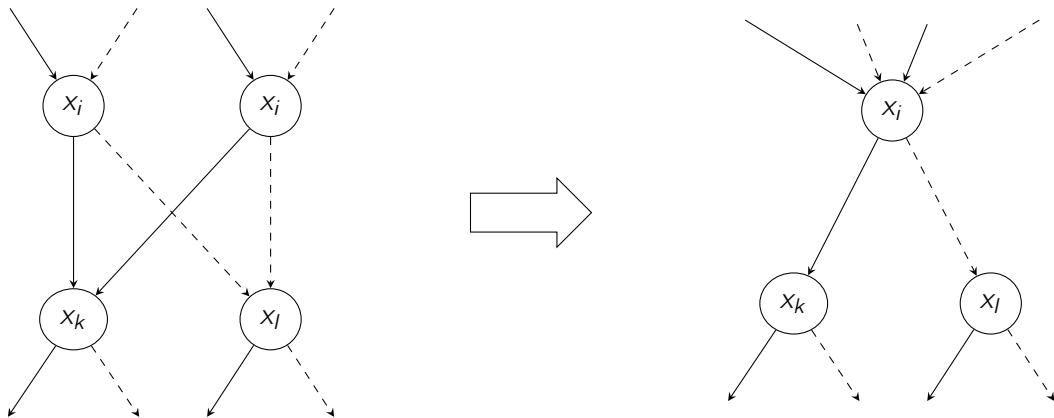
BDD - Reduction Rules

Delete Rule: Wenn beide ausgehende Kanten von x_i auf x_k zeigen, können wir x_i löschen und die eingehenden Kanten von x_i entsprechend auf x_k umlegen



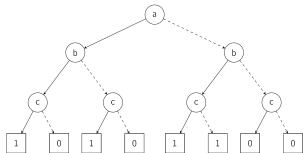
BDD - Reduction Rules

Merge Rule: Wenn wir zwei Kopien von x_i haben, die genau gleich auf x_k und x_l verweisen, können wir die Kopien von x_i mergen (zusammenlegen)

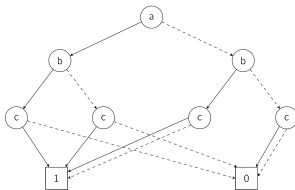


Beispiel

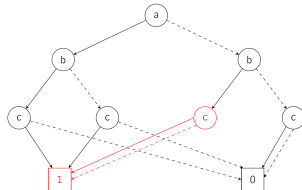
Decision Tree



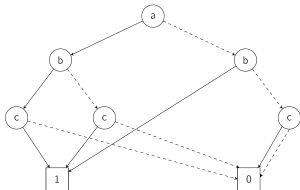
1: Nur ein 0 und 1 Knoten



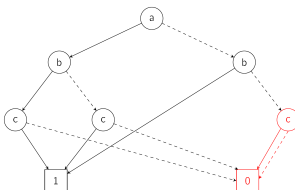
2: Delete Rule



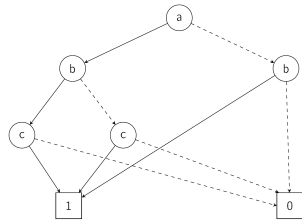
2: Delete Rule



3: Delete Rule

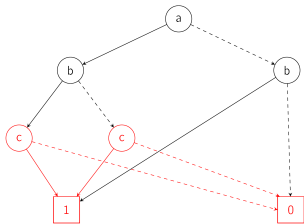


3: Delete Rule

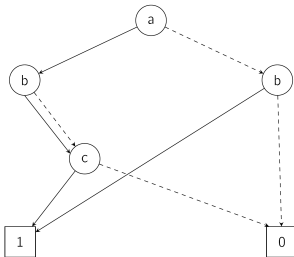


Beispiel

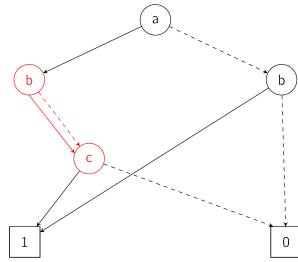
4: Merge rule



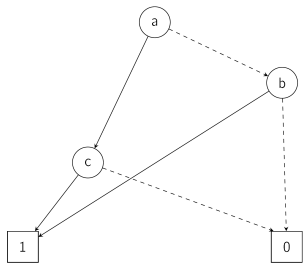
4: Merge rule



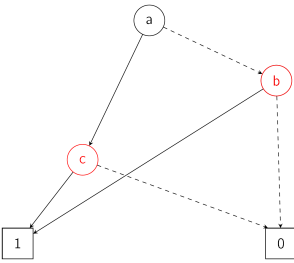
5: Delete rule



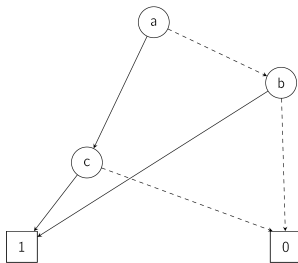
5: Delete rule



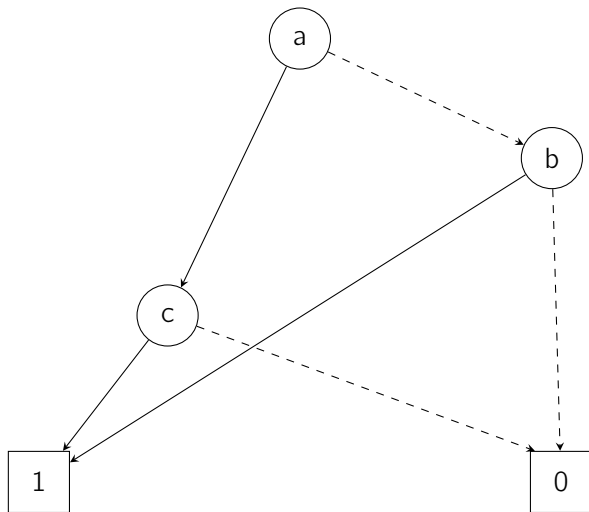
6: Merge rule?



6: Nein, weil $b \neq c$

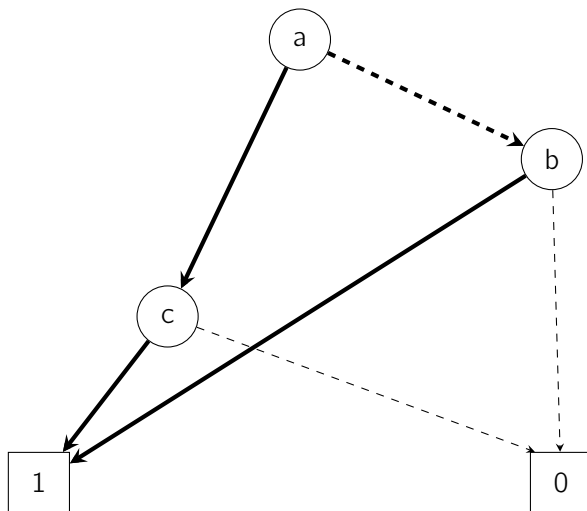


Beispiel – ITE-Form



```
if (a == true) then
  if (c == true) then 1
  else 0
else
  if (b == true) then 1
  else 0
```


Beispiel – Disjunktive Normalform



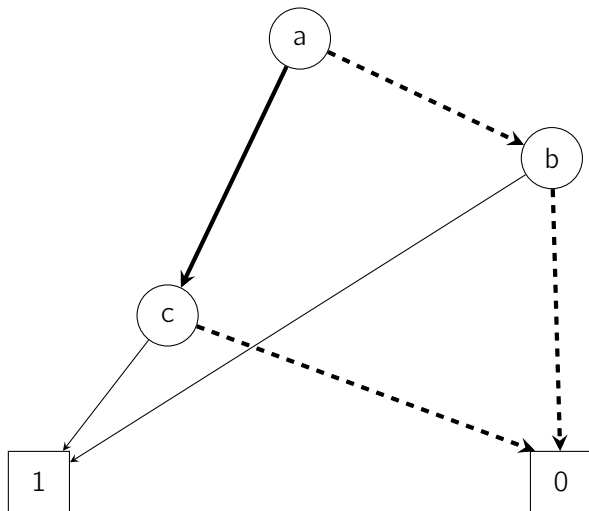
Disjunktive Normalform ablesen:

Pfade von Wurzel zu „1“

$$(a \wedge c) \vee (\neg a \wedge b)$$

Nicht notwendigerweise minimal!

Beispiel – Konjunktive Normalform



Konjunktive Normalform ablesen:

Pfade von Wurzel zu „0“

Variablen negieren

$$(\neg a \vee c) \wedge (a \vee b)$$

Nicht notwendigerweise minimal!

Vereinfachte Reduktion (Beads)

- Bisherige Reduktion im Baum durch Anwendung von
 - Merge Rule
 - Delete Rule
- Vereinfachte Version durch „Aufschreiben“ der Ergebnisse der Wahrheitstabelle als Wort der Länge 2^n , wobei n die Anzahl der Eingangsvariablen ist
 - Beispiel: $f(a, b, c) = 01011100$
- Sogenannte **Beads** (binäre Wörter der Länge 2^n)
 - Haben nicht die Form **ww**, wobei **w** ein Wort der Länge 2^{n-1} ist
 - Beispiel: 0000 0001 ist ein Bead, 1001 1001 ist kein Bead
 - Wörter, die keine Beads sind, werden als **Squares** bezeichnet
- Beispiel für $n = 2$:
 - Alle Beads der Länge 4 sind:
0001, 0010, 0011, 0100, 0110, 0111, 1000, 1001, 1011, 1100, 1101, 1110
 - Keine Beads der Länge 4 sind:
0000, 0101, 1010, 1111

Beispiel mit Beads

Beispiel von Folie 34:

| a | b | c | y |
|-----|-----|-----|-----|
| 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 |



Auslesen des binären Wortes 1010 1100 der Länge 2^3

Beobachtung:

linke Hälfte entspricht $a = 1$,
rechte Hälfte entspricht $a = 0$.

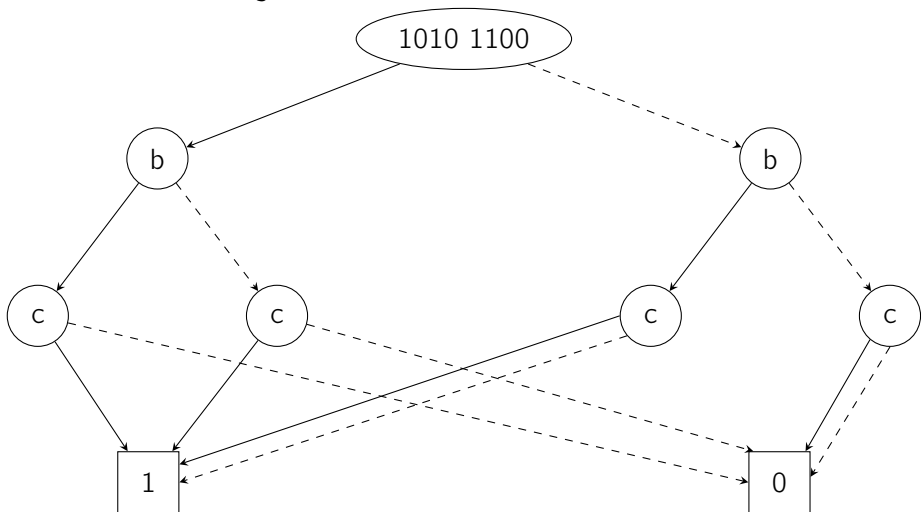
Nachfolgend zwei Beispiele:

- Entscheidungsbaum vorhanden
- Ohne Entscheidungsbaum

Beispiel mit Beads

Beispiel von Folie 34:

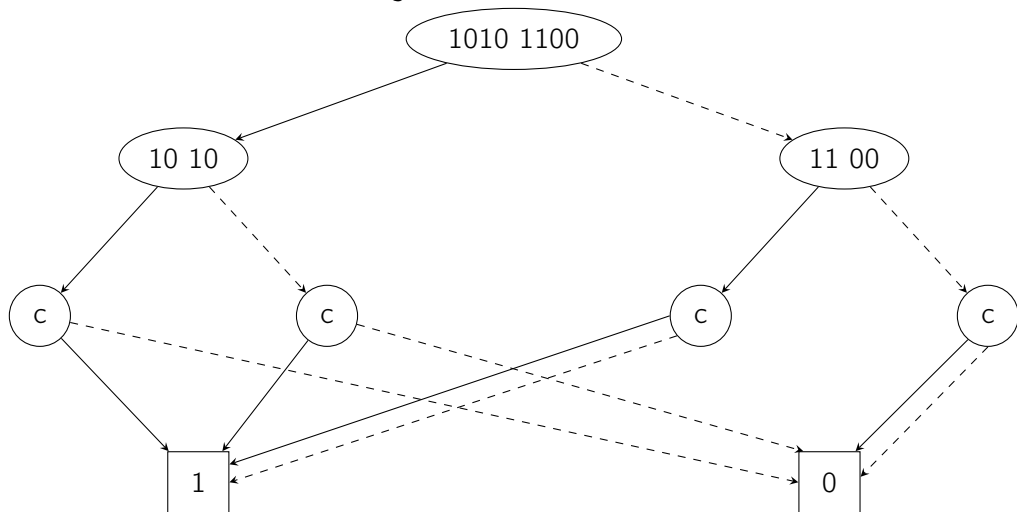
In a Wahrheitswerte eintragen



Beispiel mit Beads

Beispiel von Folie 34:

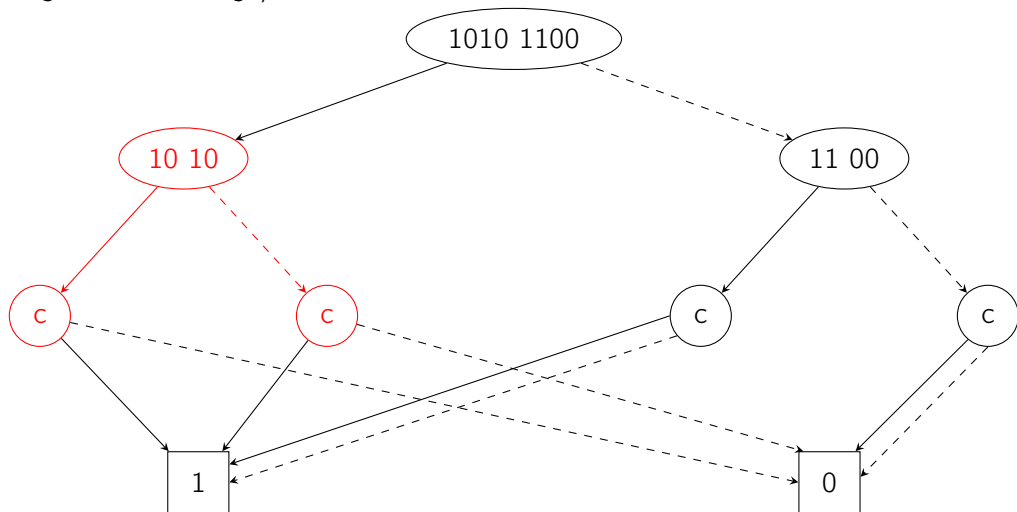
Halbieren und nach unten übertragen



Beispiel mit Beads

Beispiel von Folie 34:

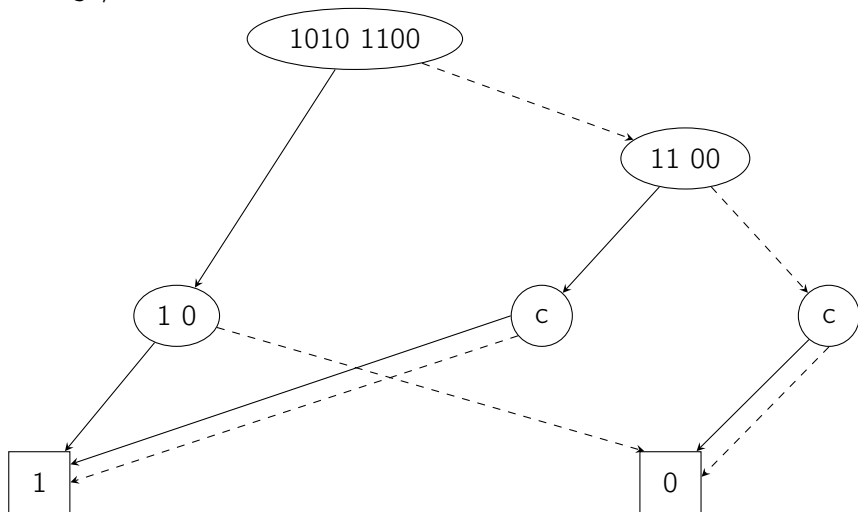
Falls gleich **ww**, Merge/Delete Rule



Beispiel mit Beads

Beispiel von Folie 34:

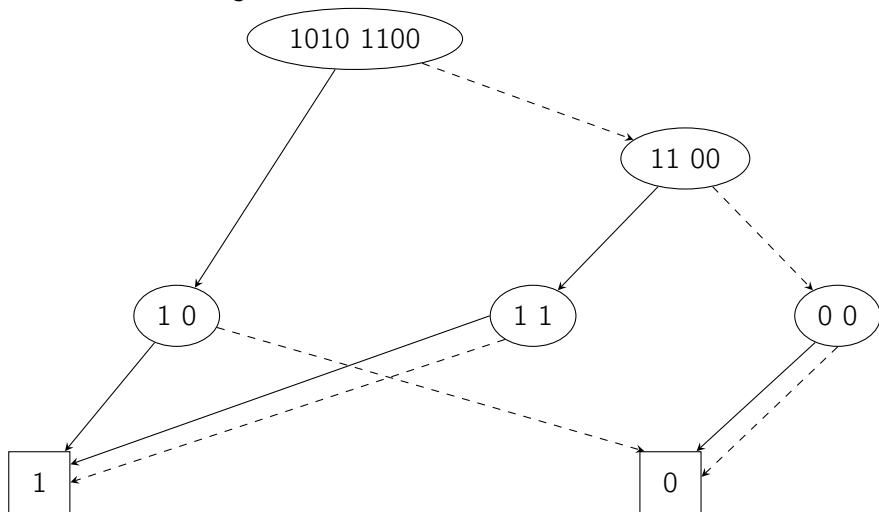
Falls gleich **ww**, Merge/Delete Rule



Beispiel mit Beads

Beispiel von Folie 34:

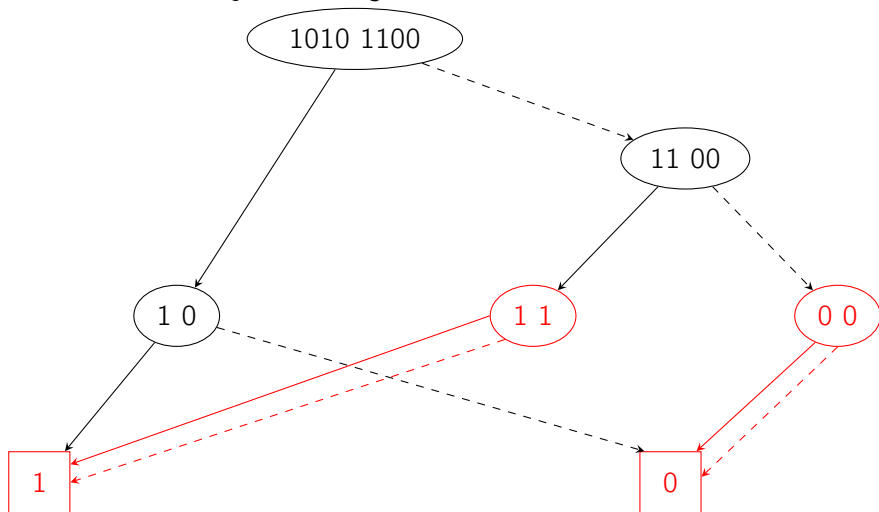
Halbieren und nach unten übertragen



Beispiel mit Beads

Beispiel von Folie 34:

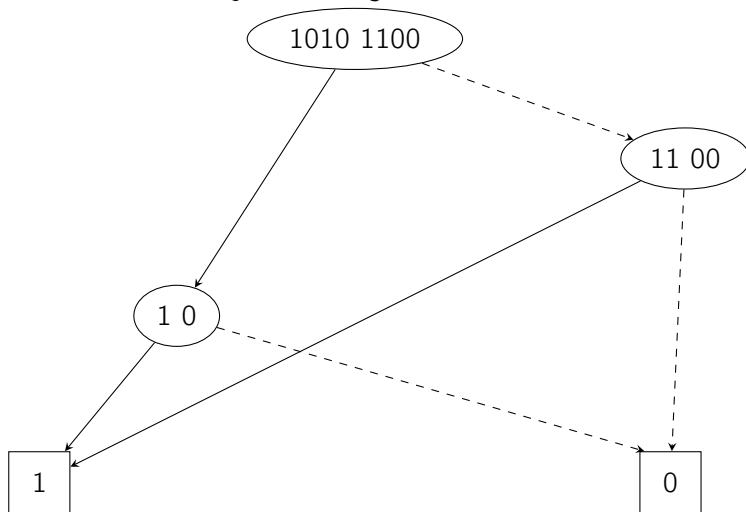
Falls gleich 00...0 oder 11...1, je nach eingehender Kante direkt auf 0 oder 1 umleiten



Beispiel mit Beads

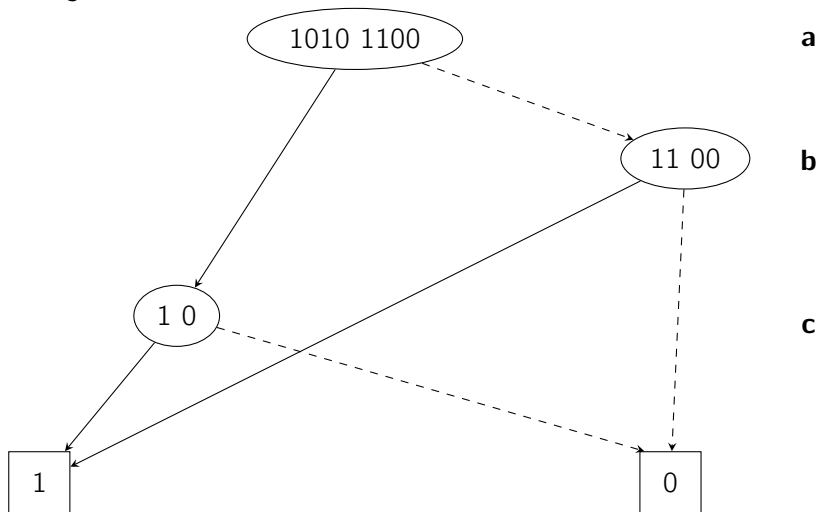
Beispiel von Folie 34:

Falls gleich $00 \dots 0$ oder $11 \dots 1$, je nach eingehender Kante direkt auf 0 oder 1 umleiten



Beispiel mit Beads

Beispiel von Folie 34:
Variablenbeschriftung!



Beispiel mit Beads ohne Decision Tree

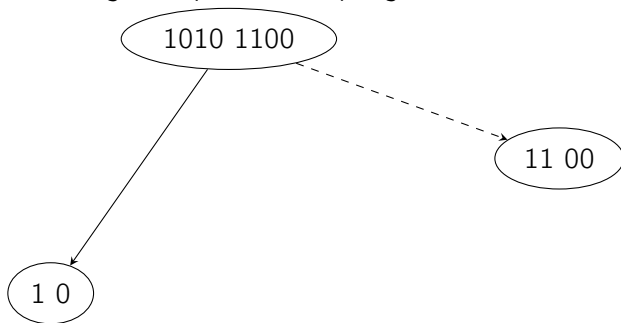
Beispiel von Folie 34:
Wahrheitswerte eintragen

1010 1100

Beispiel mit Beads ohne Decision Tree

Beispiel von Folie 34:

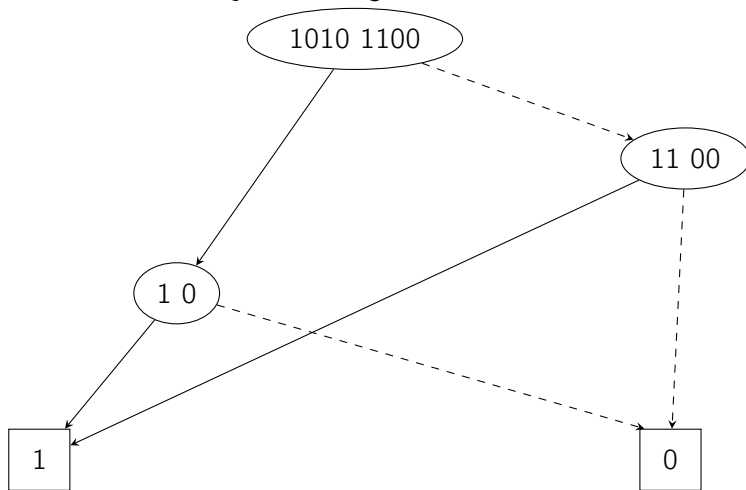
Halbieren und nach unten übertragen; Squares überspringen Ebenen, bis Beads übrig bleiben



Beispiel mit Beads ohne Decision Tree

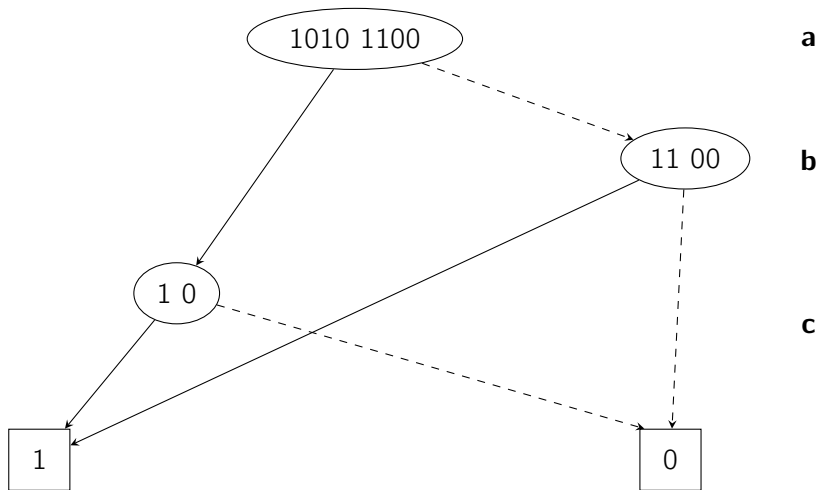
Beispiel von Folie 34:

Falls gleich 00...0 oder 11...1, je nach eingehender Kante direkt auf 0 oder 1 umleiten



Beispiel mit Beads ohne Decision Tree

Beispiel von Folie 34:

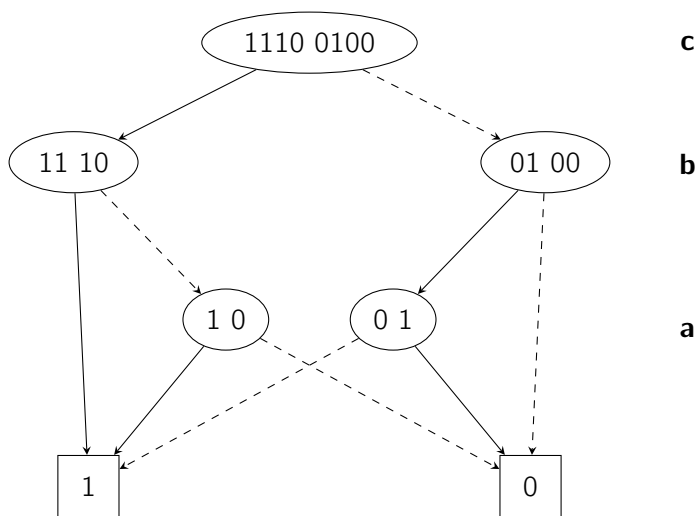


Variablenreihenfolge

Selbes Beispiel mit anderer Variablenreihenfolge:

| <i>c</i> | <i>b</i> | <i>a</i> | <i>y</i> |
|----------|----------|----------|----------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Benötigt mehr Knoten!



- Größe von BDDs hängt von Variablenreihenfolge in der Wahrheitstabelle ab
- Für n Variablen gibt es $n!$ unterschiedliche Permutationen
 - Für $n = 10$: $n! > 3.6 \cdot 10^6$
 - Für $n = 20$: $n! > 2.4 \cdot 10^{18}$
 - ⇒ Wächst exponentiell in der Anzahl der Variablen
- Minimale Form zu finden ist schwer

- Es gibt Boolesche Funktionen, sodass unabhängig von der Variablenreihenfolge alle BDDs exponentielle Größe haben
- Die meisten praktisch relevanten Funktionen haben geringe Größe
- Bekannte schwer zu vereinfachende Funktion: *Multiplikation*
 - Unabhängig von der Variablenordnung exponentielle Größe der Vereinfachung
- Für manche exponentiell große disjunktive Normalform gibt es BDDs in polynomieller Größe: z.B.: *Majority-Funktion*
 - Funktion in n Variablen
 - Liefert false, wenn mehr als die Hälfte der Variablen den Wert false haben, ansonsten true

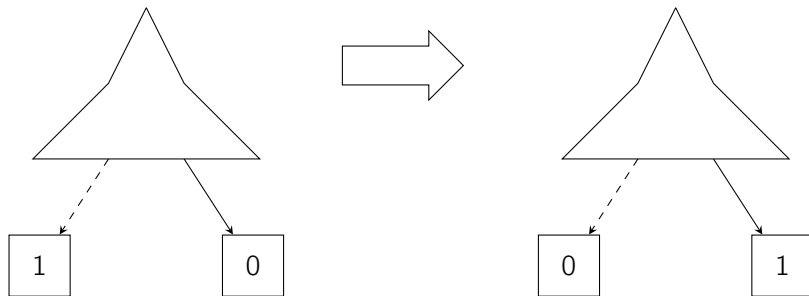
Eigenschaften von BDDs

- In einem BDD kommen **alle** „Beads der Wahrheitstabelle“ und **nur diese** vor (keine Squares)
- Darstellung einer bestimmten Booleschen Funktion ist bei gegebener Variablenreihenfolge **isomorph**
- Zwei isomorphe BDDs beschreiben äquivalente Boolesche Ausdrücke
- Typische Funktionen sind einfach mittels BDD-Darstellung zu realisieren
- Operationen können direkt auf der kompakten Darstellung durchgeführt werden

Typische Funktionen auf BDDs

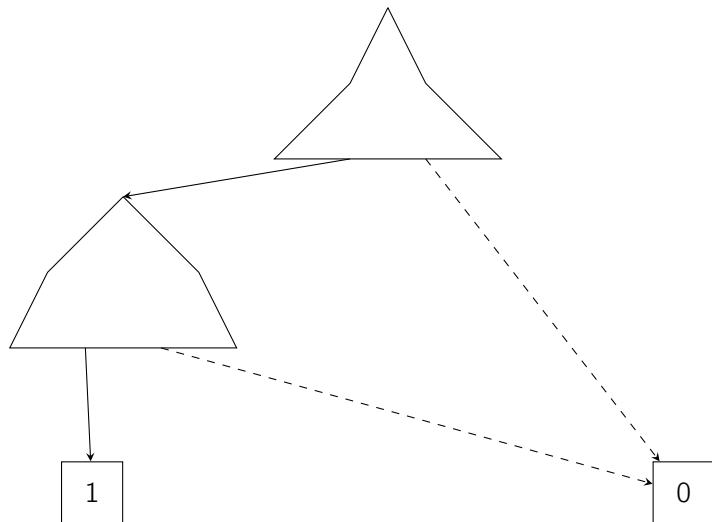
Negation

1 und 0 vertauschen



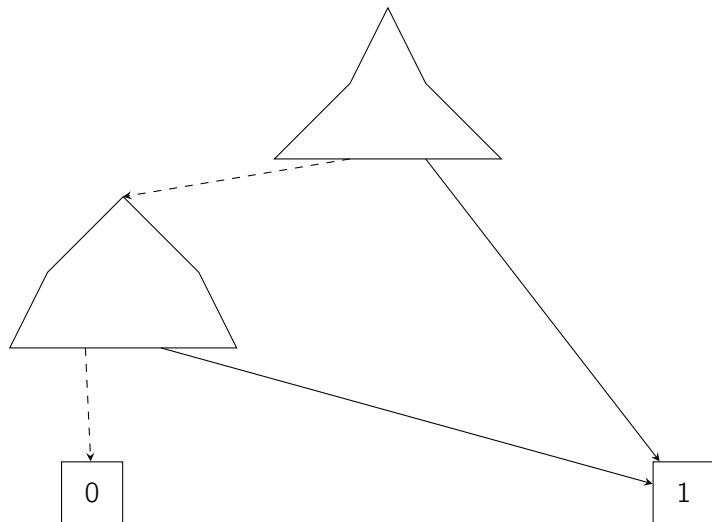
Typische Funktionen auf BDDs

UND



Typische Funktionen auf BDDs

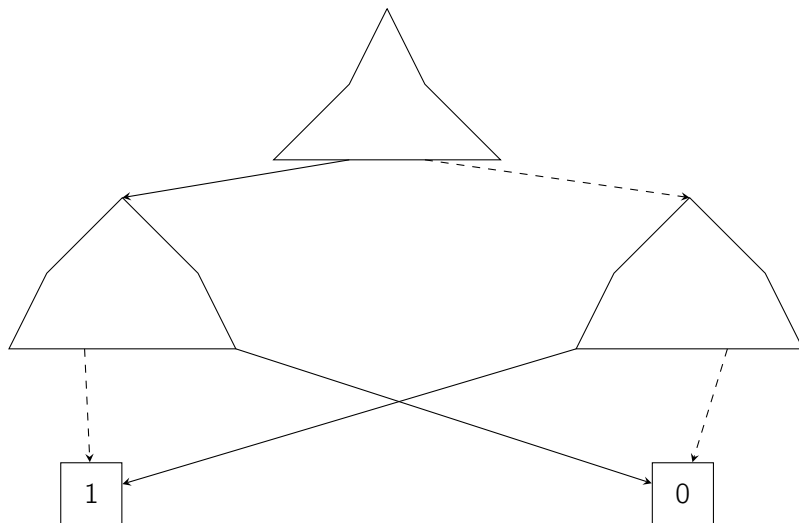
ODER



Typische Funktionen auf BDDs

XOR

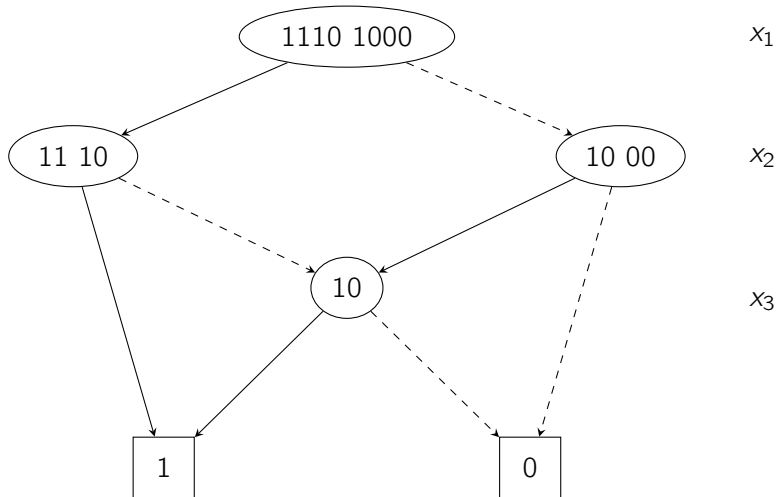
einen Operanden duplizieren



Beispiel

Majority-Funktion in drei Variablen

| x_1 | x_2 | x_3 | y |
|-------|-------|-------|-----|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



- Eine Boolesche Funktion, die k „X“ enthält steht eigentlich für 2^k unterschiedliche Funktionen, da jedes „X“ die Werte „0“ oder „1“ annehmen kann
- Für Funktionen in vielen Variablen und mit vielen „X“ sind das sehr viele unterschiedliche Funktionen, die man alle vereinfachen müsste, um die „kleinste“ Funktion zu finden
- Daher: Don't Cares in BDDs integrieren

Beispiel

Halbierte Beads
untereinander
schreiben

11X0

X101



11X0 X101

Beispiel

Halbierte Beads
untereinander
schreiben

11X0

X101

Kann man durch Setzen der „X“ ein Square erhalten?

11X0 X101

Beispiel

Halbierte Beads
untereinander
schreiben

11X0

X101

Kann man durch Setzen der „X“ ein Square erhalten?

Nein!



11X0 X101

Beispiel

Halbierte Beads
untereinander
schreiben

11X0

X101

Kann man durch Setzen der „X“ ein Square erhalten?

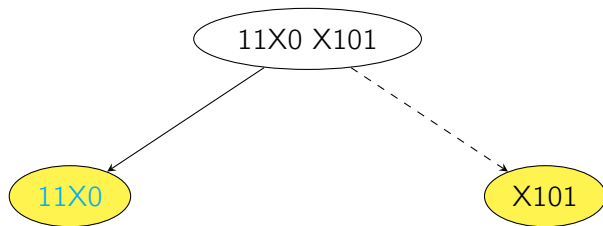
Nein!

Daher halbierte Beads in die nächste Ebene.

11X0 X101

Beispiel

11
X0

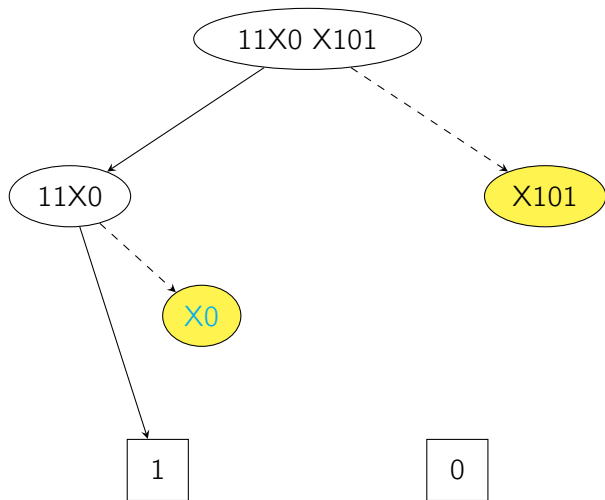


1

0

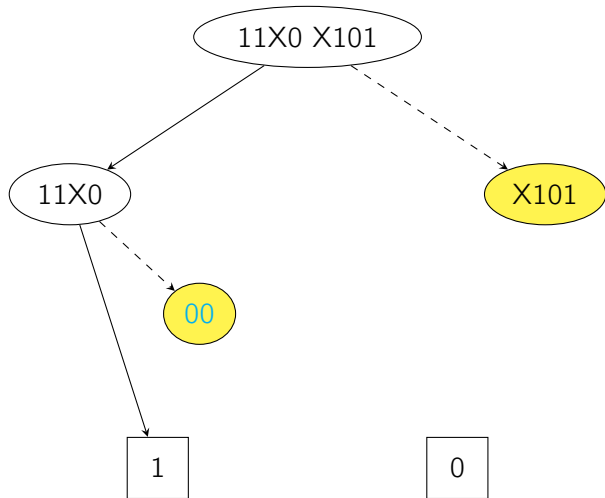
Beispiel

X
0

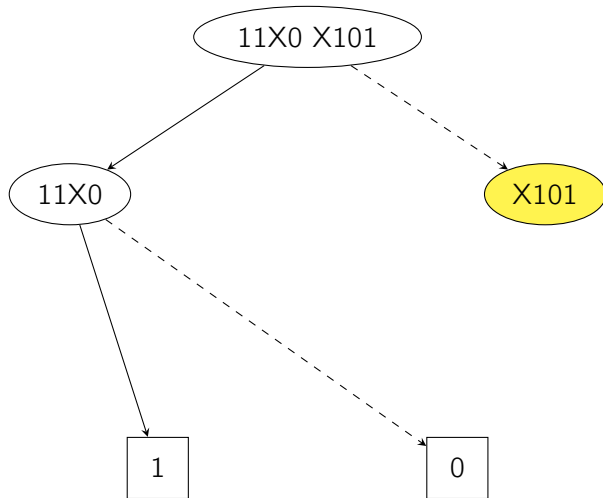


Beispiel

0
0

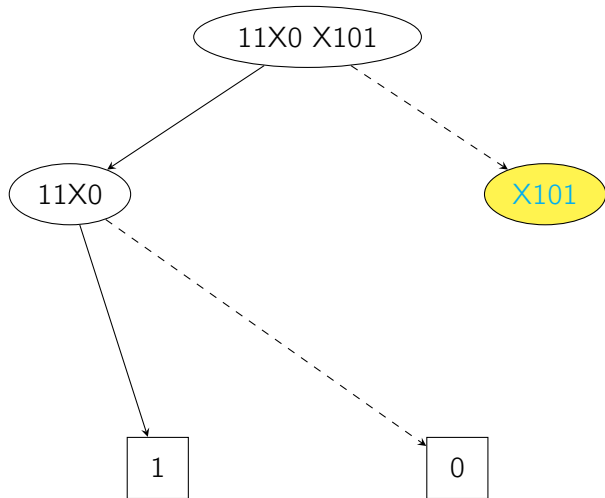


Beispiel



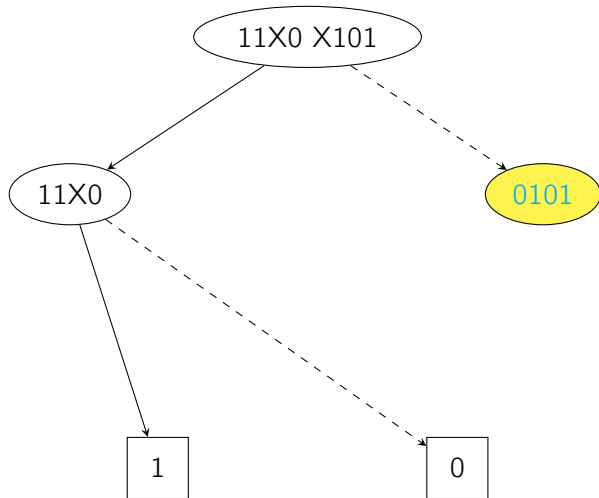
Beispiel

X1
01

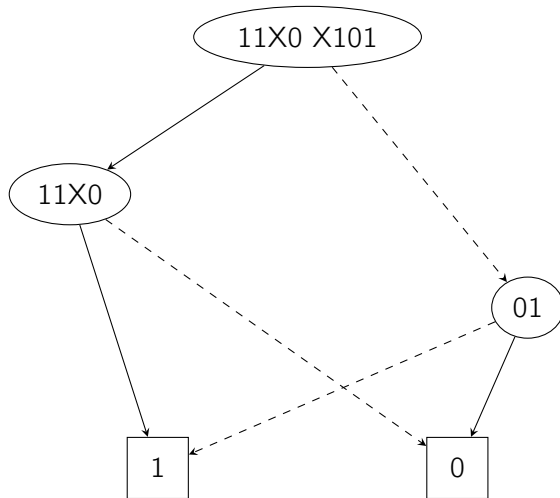


Beispiel

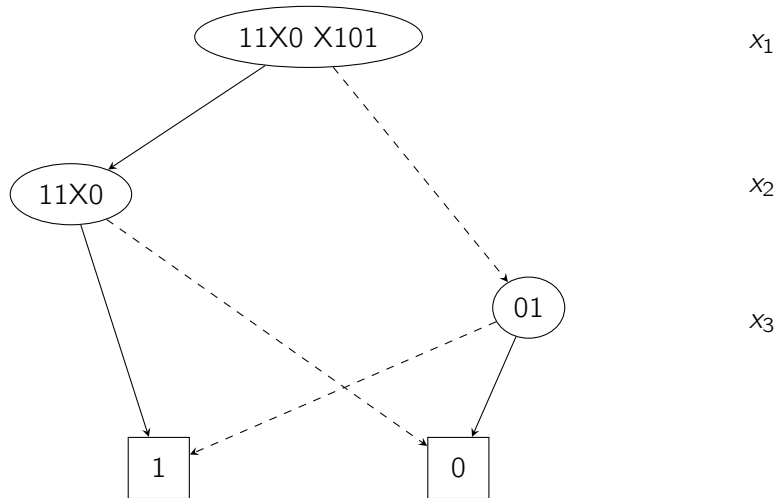
01
01



Beispiel



Beispiel



Noch ein Beispiel

01X1

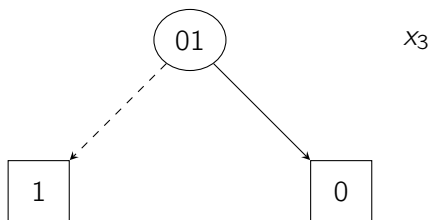
0X01

0101

0101

01X1 0X01

0101



Fixieren aufschieben

- Ein abschließendes Beispiel soll zeigen, dass es manchmal gut ist, die Werte von „Don't Care“-Stellen erst bei weiter unten liegenden Ebenen zu fixieren

Zwei Varianten:

- Gierige Variante: X so schnell als möglich durch 0 oder 1 ersetzen
- Geduldige Variante: X so spät als möglich durch 0 oder 1 ersetzen

Beispiel (gierige Variante)

X00X
1XX1

X00X 1XX1

1

0

Beispiel (gierige Variante)

1001
1001

1001 1001

1

0

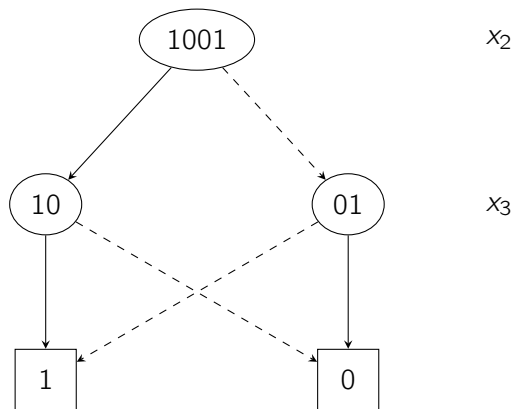
Beispiel (gierige Variante)

1001

1

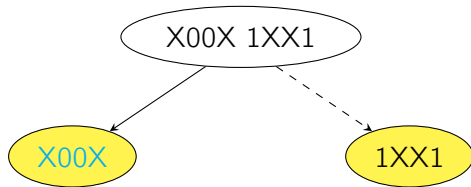
0

Beispiel (gierige Variante)



Beispiel (geduldige Variante)

X0
0X

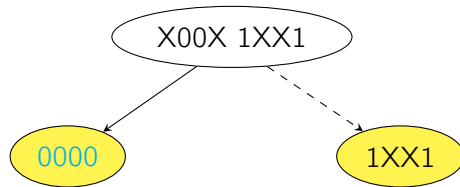


1

0

Beispiel (geduldige Variante)

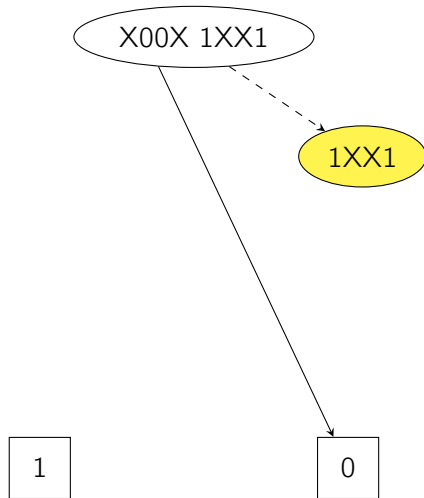
00
00



1

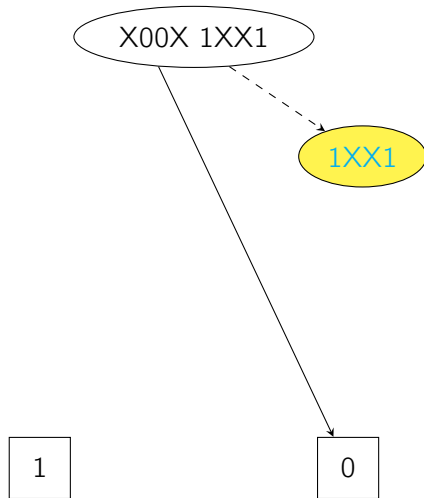
0

Beispiel (geduldige Variante)



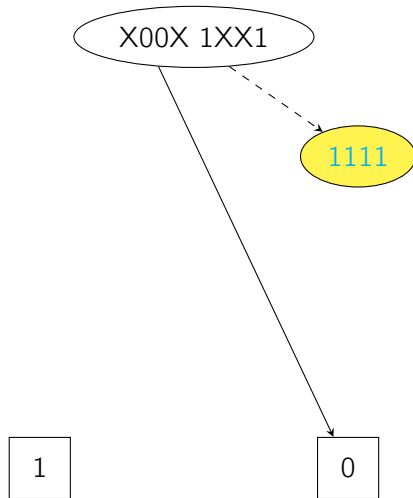
Beispiel (geduldige Variante)

1X
X1

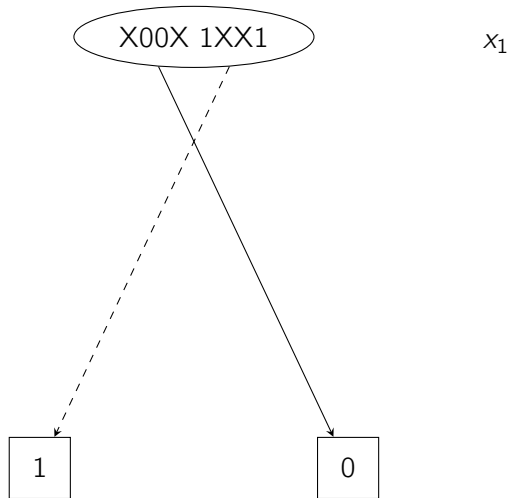


Beispiel (geduldige Variante)

11
11



Beispiel (geduldige Variante)



- Das voran gegangene Beispiel hat gezeigt, dass es manchmal möglich ist, eine bessere Vereinfachung zu erhalten, wenn man „Don't Cares“ „überleben“ lässt und erst später deren Werte fixiert
- Man benötigt also gute Heuristiken für „Don't Care“-Vereinfachungen, falls man BDDs mit „Don't Cares“ automatisiert minimieren will

- Boolesche Algebra
 - Algebra der Logik
 - Methode um boolesche Funktionen darzustellen
- Repräsentation Boolescher Funktionen
 - Formel
 - Wahrheitstabelle
 - If-Then-Else (ITE) Form
 - Binäre Entscheidungsdiagramme (BDDs)