technische universität
dortmund

Rechnerstrukturen WS 2012/13

- Boolesche Funktionen und Schaltnetze
  - Repräsentationen boolescher Funktionen (Wiederholung)
  - Normalformen boolescher Funktionen (Wiederholung)
  - Repräsentation boolescher Funktionen mit OBDDs
  - Synthese von OBDDs für boolesche Funktionen
  - Schaltnetze

Hinweis: *Folien teilweise a. d. Basis von Materialien von Thomas Jansen*

23. Oktober 2012

Repräsentationen boolescher Funktionen

Definition 4

Seien $n, m \in \mathbb{N}$. Eine Funktion $f : B^n \to B^m$ heißt boolesche Funktion.

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

Anzahl boolescher Funktionen

boolesche Funktion $f : B^n \to B^m$ als Wertetabelle darstellbar
mit $|B^n| = 2^n$ Zeilen
und $|B^m| = 2^m$ Möglichkeiten je Zeile

$\Rightarrow 2^{m^{2^n}} = 2^{m \cdot 2^n}$ boolesche Funktionen $f : B^n \to B^m$

Beispiel $2^{1 \cdot 2^2} = 2^4 = 16$ boolesche Funktionen $f : B^2 \to B$

Alle booleschen Funktionen $f : B^2 \to B$

x	0	0	1	1			x	0	0	1	1		
y	0	1	0	1			y	0	1	0	1		
f_1	0	0	0	0	Nullfkt.	0	f_9	1	0	0	0	NOR	
f_2	0	0	0	1	AND	$\wedge$	f_{10}	1	0	0	1	Äquiv.	$\Leftrightarrow$
f_3	0	0	1	0			f_{11}	1	0	1	0	Negation	$\neg y$
f_4	0	0	1	1	Proj.	x	f_{12}	1	0	1	1		
f_5	0	1	0	0			f_{13}	1	1	0	0	Negation	$\neg x$
f_6	0	1	0	1	Proj.	y	f_{14}	1	1	0	1	Impl.	$\Rightarrow$
f_7	0	1	1	0	XOR	$\oplus$	f_{15}	1	1	1	0	NAND	
f_8	0	1	1	1	OR	$\vee$	f_{16}	1	1	1	1	Einsfkt.	1

Verwenden im Weiteren $\wedge$ (Konjunktion), $\vee$ (Disjunktion), $\bar{}$ (Negation)

Darstellung boolescher Funktionen

gerade gesehen Wertetabelle
(Orientierung meistens wie hier)

x	y	f_7
0	0	0
0	1	1
1	0	1
1	1	0

bei fester Reihenfolge Wertevektor

$f_7 : (0, 1, 1, 0)$

Index und Minterm

Index	x	y	f_7	
0	0	0	0	nicht einschlägig
1	0	1	1	einschlägig
2	1	0	1	einschlägig
3	1	1	0	nicht einschlägig

Definition

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

Ein Minterm ist nur mit Negationen und Konjunktionen darstellbar:

$$x_j = \begin{cases} 0 & \rightsquigarrow \overline{x_j} \\ 1 & \rightsquigarrow x_j \end{cases}$$

und dann Konjunktion all dieser Literale ($\hat{=}$ [negierte] Variable)

Normalformen

Definition 8

- Die Darstellung von f als Disjunktion all ihrer Minterme zu einschlägigen Indizes heißt disjunktive Normalform (DNF).
- Die Darstellung von f als XOR-Verknüpfung all ihrer Minterme zu einschlägigen Indizes heißt Ringsummen-Normalform (RNF).

Anmerkung Normalformen sind eindeutig.

Beispiel

Index	x	y	f_7	Minterm
0	0	0	0	$\overline{x}\,\overline{y}$
1	0	1	1	$\overline{x}\,y$
2	1	0	1	$x\,\overline{y}$
3	1	1	0	$x\,y$

DNF von f_7 $\overline{x}\,y \vee x\,\overline{y}$

RNF von f_7 $\overline{x}\,y \oplus x\,\overline{y}$

Funktionale Vollständigkeit

Beobachtung jede boolesche Funktion $f : B^n \to B$ nur mittels Konjunktion, Disjunktion und Negation darstellbar (z. B. durch ihre DNF)

Definition 5

Eine Menge $\mathcal{F}$ von booleschen Funktionen heißt funktional vollständig, wenn sich jede boolesche Funktion durch Einsetzen und Komposition von Funktionen aus $\mathcal{F}$ darstellen lässt.

Satz 6

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

Darstellungen 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

Eine Datenstruktur für boolesche Funktionen

Ziel $f : B^n \to B$ darstellen

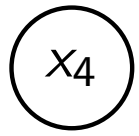
Wünsche

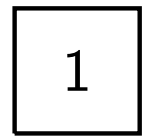
- zu einer Belegung $x_1, x_2, \ldots, x_n$ schnell den Funktionswert $f(x_1, x_2, \ldots, x_n)$ ausrechnen können
- Funktionen schnell auf Gleichheit testen können
- Funktionen schnell manipulieren (z. B. eine Variable konstant setzen) können
- schnell eine Null-Eingabe/eine Eins-Eingabe finden können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

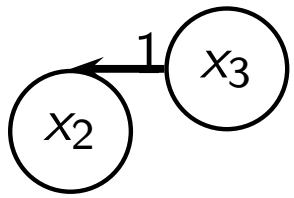
Ordered Binary Decision Diagrams

OBDDs

erster Schritt Festlegen einer Variablenordnung π
(z. B. $\pi = (x_3, x_1, x_2, x_4)$)

dann Baue πOBDD aus Knoten 

oder 

und Kanten 

nach folgenden 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

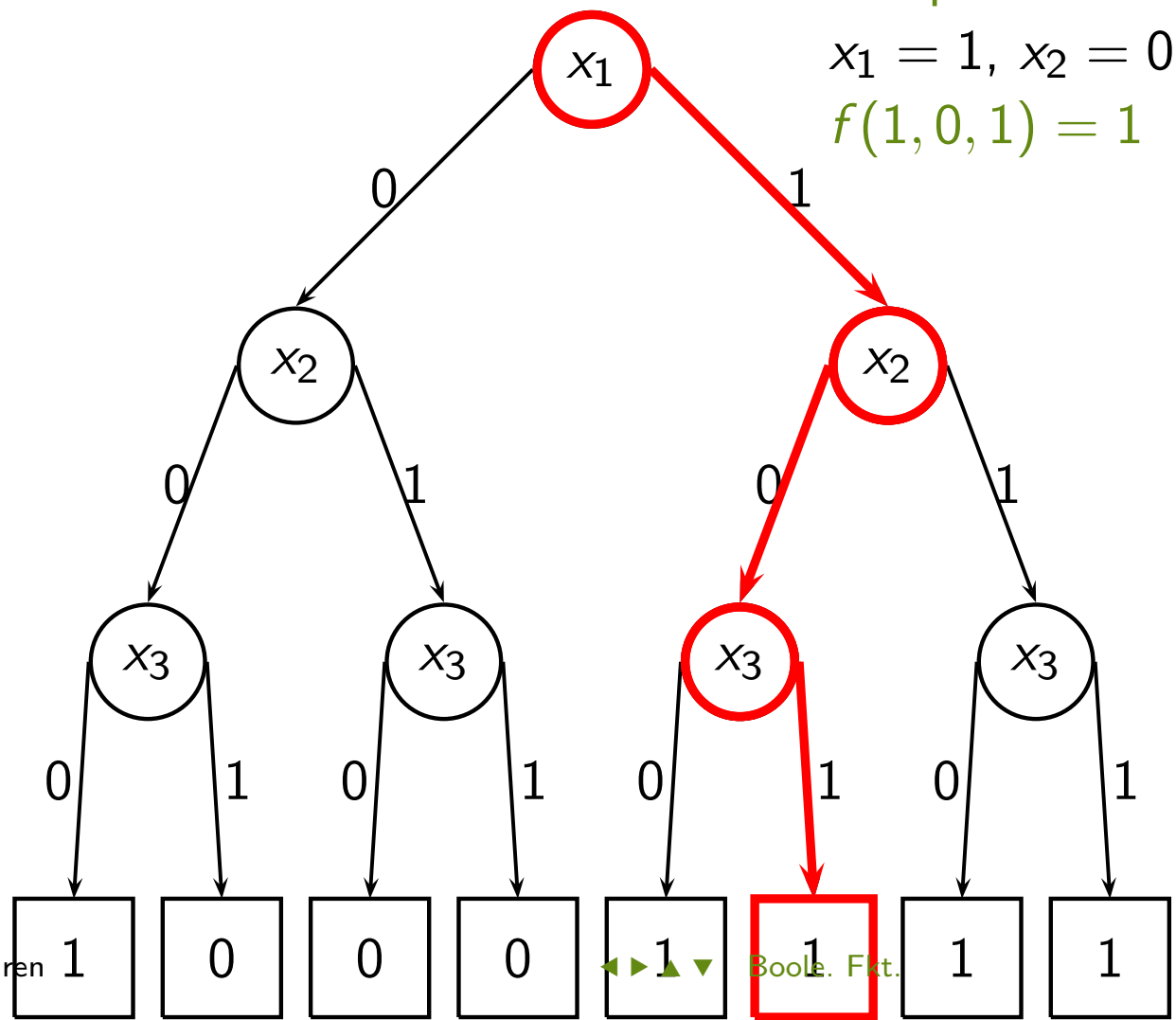
πOBDD – Ein Beispiel

Variablenordnung $\pi = (x_1, x_2, x_3)$

Beispiel Auswertung $f(1,0,1)$

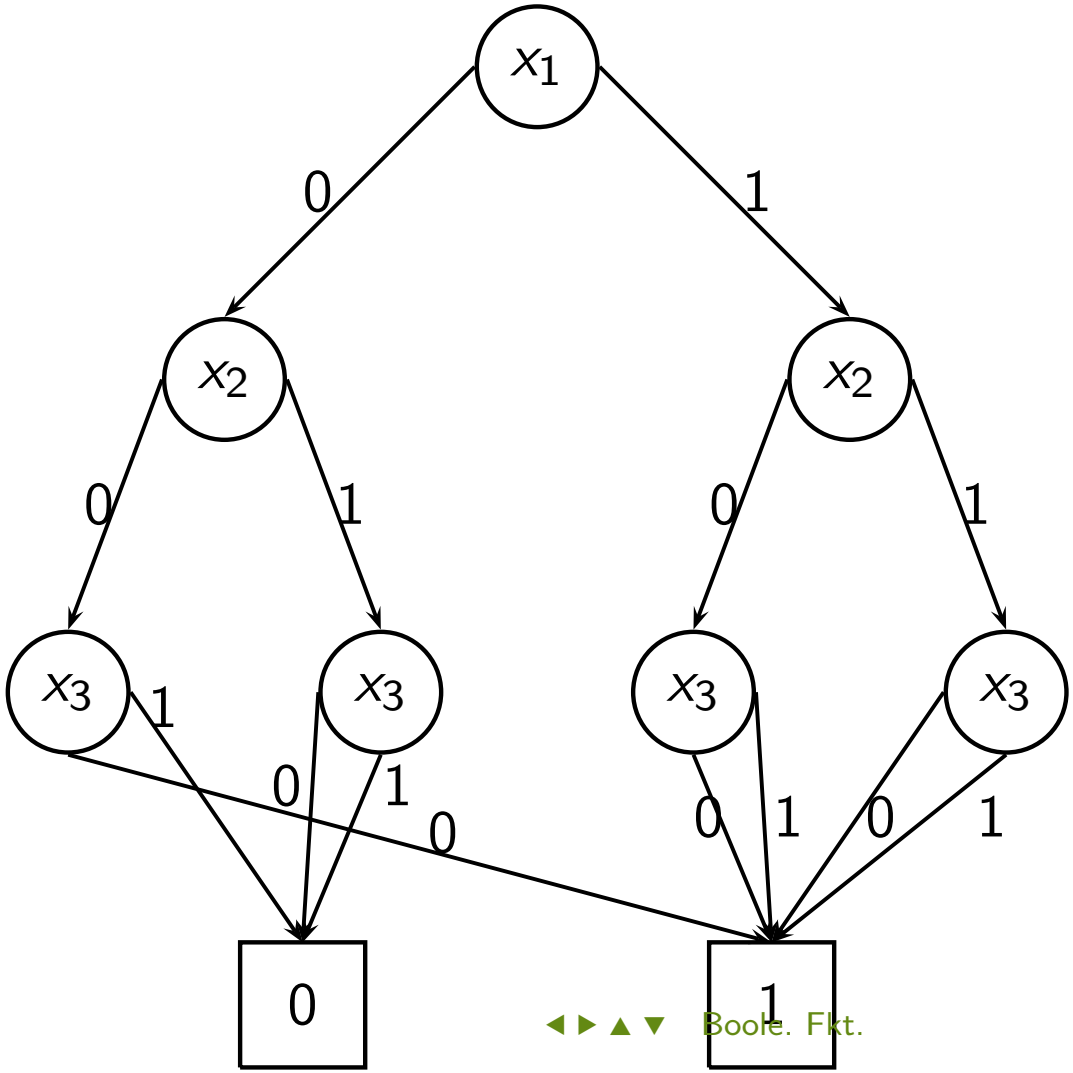
$x_1 = 1,\ x_2 = 0,\ x_3 = 1$

$f(1,0,1) = 1$



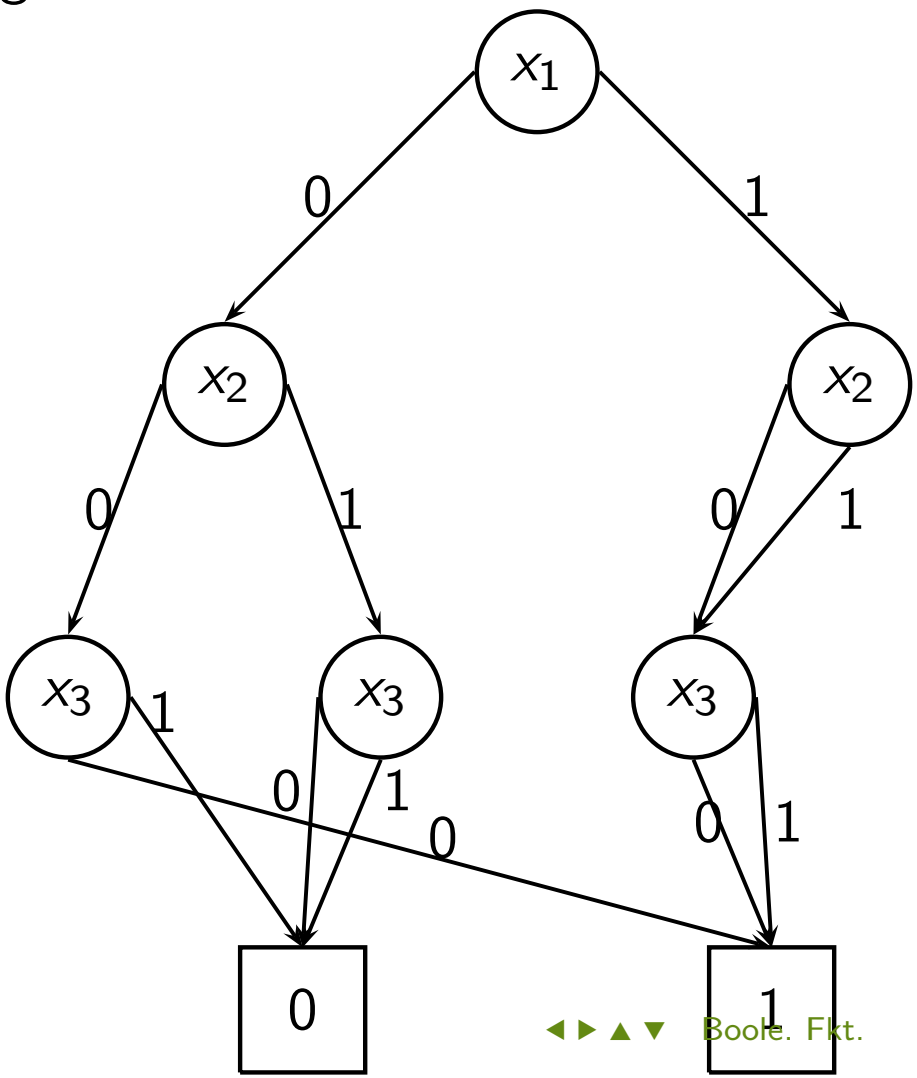
πOBDD-Größe

gleichartige Senken verschmelzen



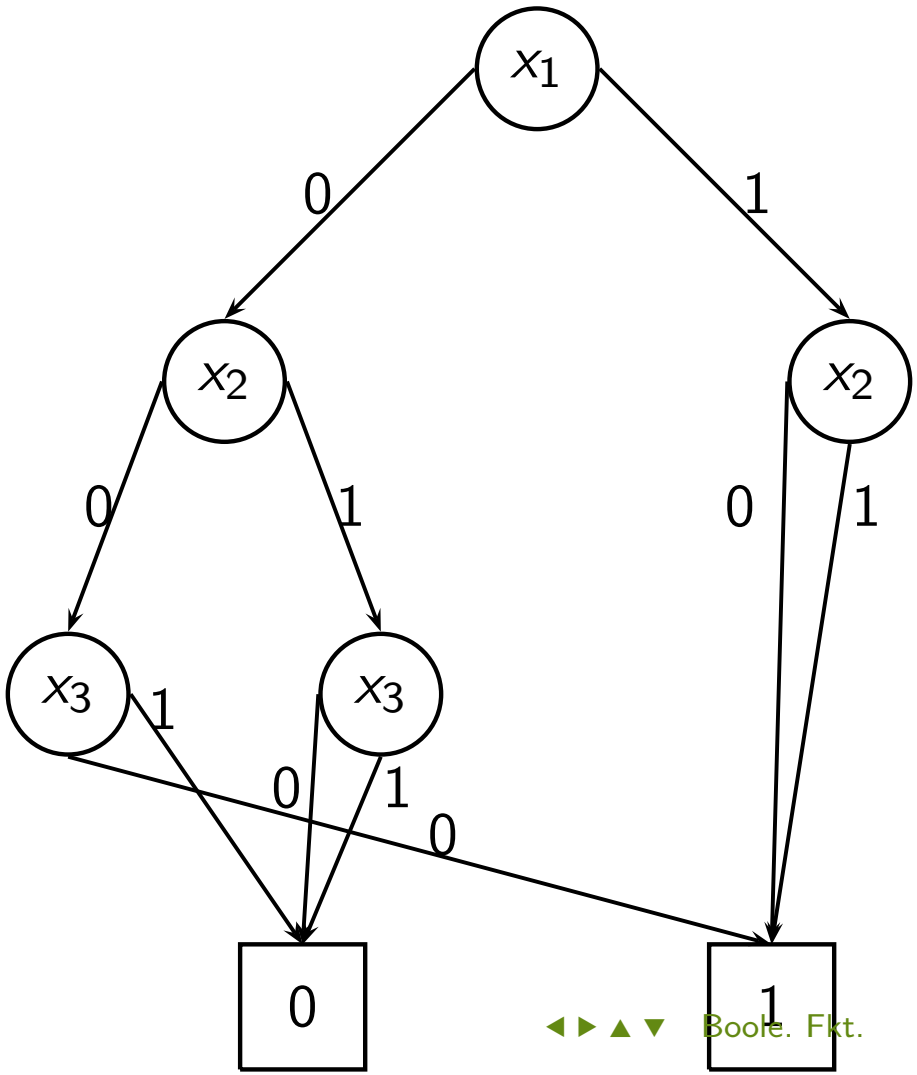
πOBDD-Größe

gleichartige Knoten verschmelzen



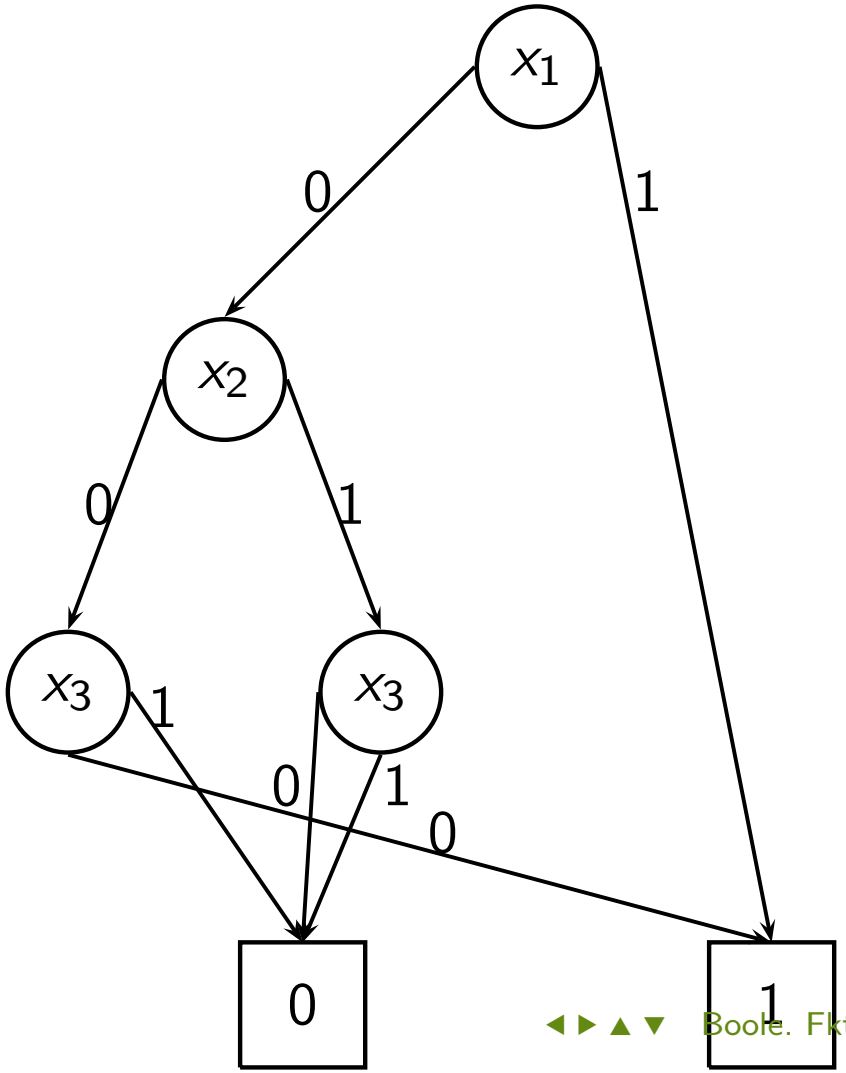
πOBDD-Größe

Knoten ohne Einfluss eliminieren



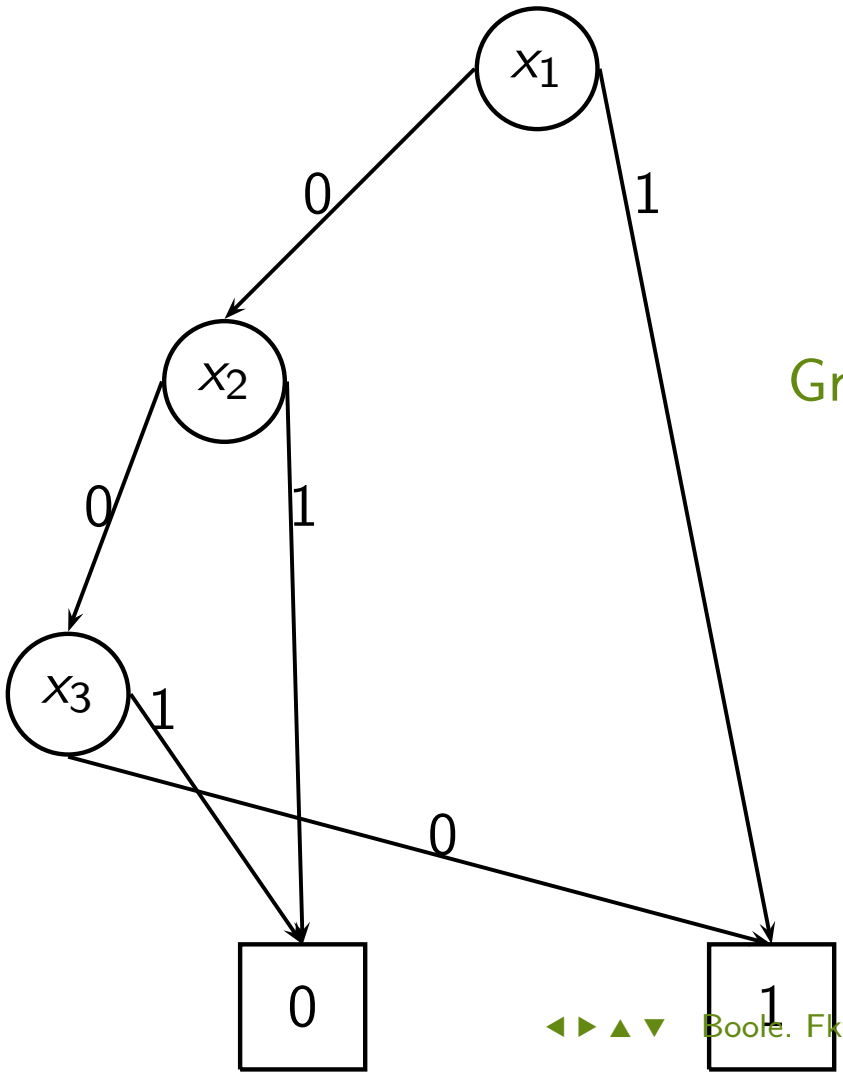
πOBDD-Größe

Knoten ohne Einfluss eliminieren



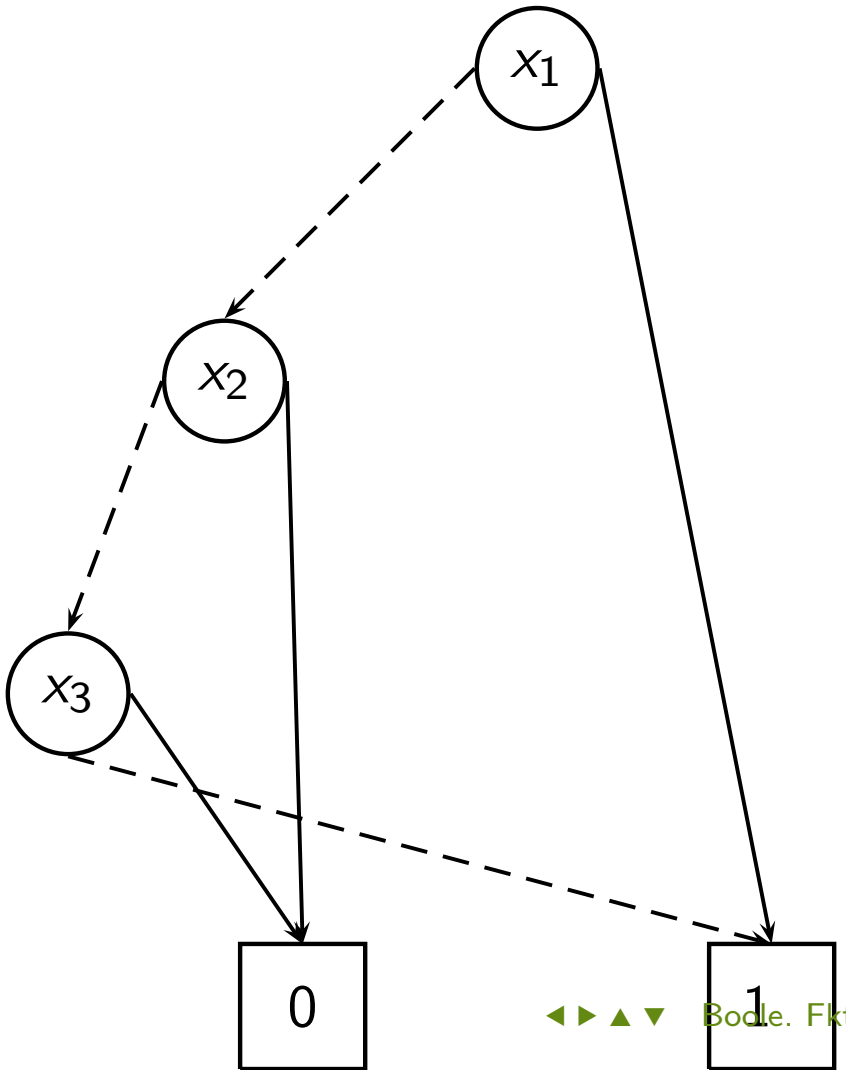
πOBDD-Größe

Knoten ohne Einfluss eliminieren



Größe minimal

Alternative Darstellung eines πOBDDs



OBDD-Reduzierung

Satz 9

Die erschöpfende Anwendung der

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

in beliebiger Reihenfolge führt zum reduzierten πOBDD.

reduziert = minimale Größe und eindeutig

Erzeugung von OBDDs

Wie kommt man zu einem πOBDD? für $f : B^n \to B$?

Kommt darauf an, wie f gegeben ist...

Welche Formate kennen wir?

- Funktionstabelle
- Wertevektor
- boolescher Ausdruck
- Normalformen
- informale Beschreibung
- OBDD
- ...

Erzeugung von OBDDs

Man will nicht irgendein πOBDD,
man will das reduzierte πOBDD.

Welchen Weg kennen wir, das zu bekommen?

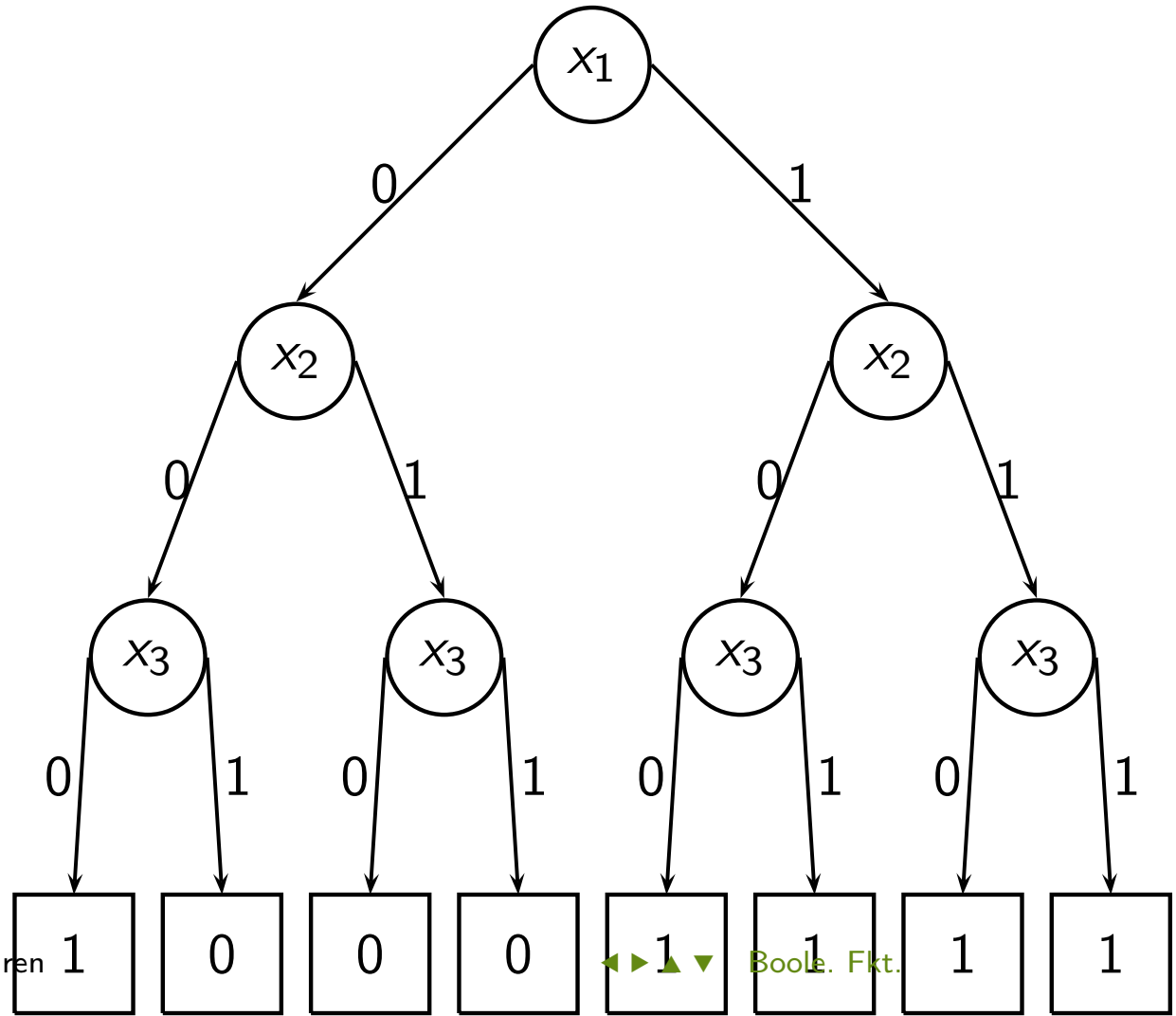
1. Erstelle vollständigen binären Baum über den Variablen, schreibe passende Funktionswerte an die Senken.
2. Reduziere.

Ist das ein vernünftiges Vorgehen?

Nur, wenn vollständiger Binärbaum nicht wesentlich größer als Eingabe und reduziertes πOBDD!

πOBDD aus vollständigem Binärbaum

Variablenordnung $\pi = (x_1, x_2, x_3)$



Größe von Repräsentationen boolescher Funktionen

Wie groß ist ein πOBDD für f, das vollständiger Binärbaum ist?

Beobachtung i-te Ebene 2^{i-1} Knoten

also $$\text{Senken} + \sum_{i=1}^{n} 2^{i-1} = \text{Senken} + \sum_{i=0}^{n-1} 2^{i} = 2^n - 1 + \text{Senken}$$

Für welche Eingabeformate ist das also akzeptabel?

sicher nur für Wertetabelle und Wertevektor

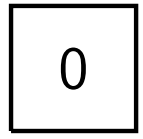
also Vorgehen fast immer nicht akzeptabel

Erkenntnis Wir brauchen eine Alternative.

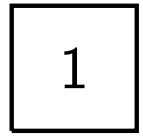
Schrittweise OBDD-Konstruktion

Idee baue πOBDD schrittweise aus „einfachsten“ πOBDDs zusammen

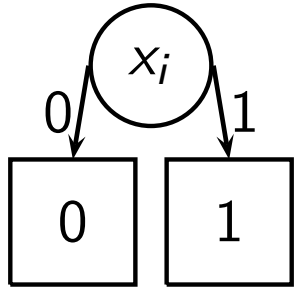
klar πOBDD für Nullfunktion



klar πOBDD für Einsfunktion



klar πOBDD für x_i



klar πOBDD für $\neg f$ aus πOBDD für f durch Invertierung der Senkenmarkierungen

OBDD-Synthese

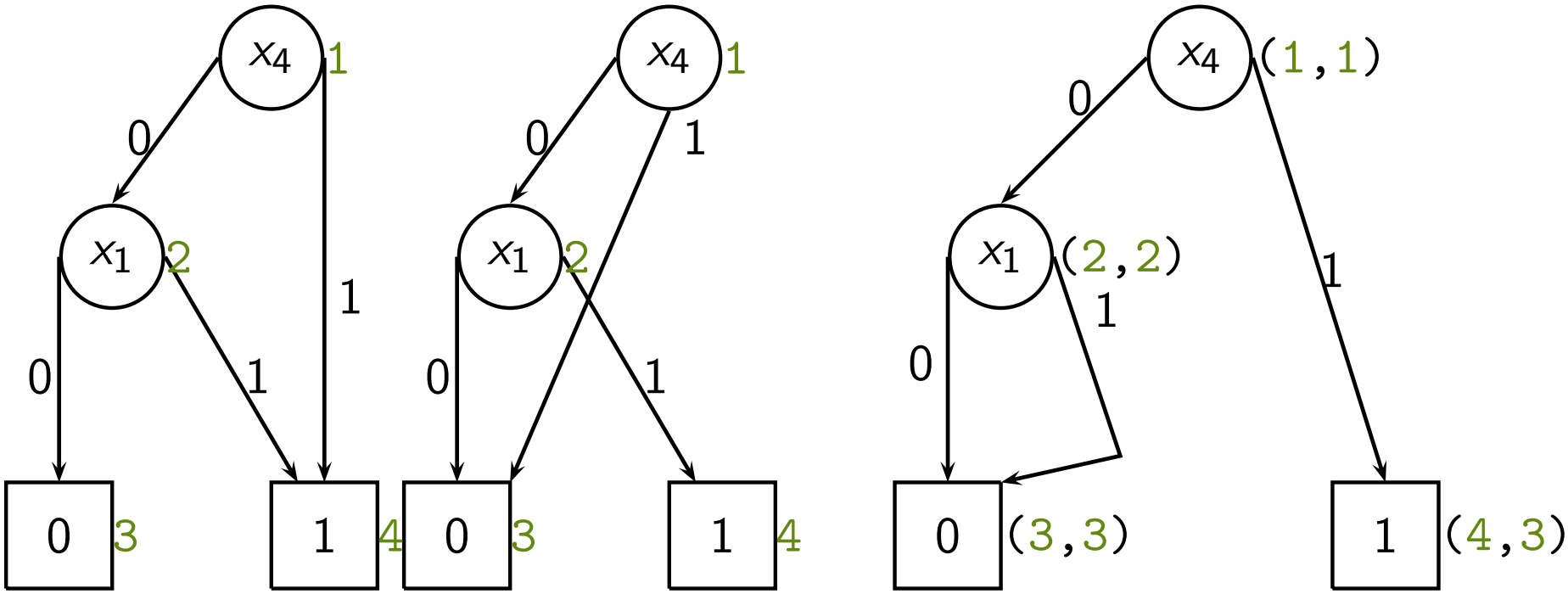
$$\left.\begin{array}{l}\pi\text{OBDD } G_1 \text{ für } f_1 : B^n \to B \\ \pi\text{OBDD } G_2 \text{ für } f_2 : B^n \to B\end{array}\right\} \rightsquigarrow \begin{array}{l}\pi\text{OBDD } G \\ \text{für } f_1 \otimes f_2\end{array}$$

für beliebige boolesche Funktion $\otimes : B^2 \to B$

Wichtig gleiche Variablenordnung π

Idee Durchlaufe beide πOBDDs parallel.

Ein einfaches Beispiel



$x_4 \vee \overline{x_4}\, x_1$ $\qquad$ $\overline{x_4}\, x_1$ $\qquad$ $(x_4 \vee \overline{x_4}\, x_1) \oplus (\overline{x_4}\, x_1) = x_4$

Variablenordnung x_4, x_2, x_1, x_3

boolesche Verknüpfung $\oplus$

πOBDD-Synthese

$$\left.\begin{array}{l} \pi\text{OBDD } G_1 \text{ für } f_1 : B^n \to B \\ \pi\text{OBDD } G_2 \text{ für } f_2 : B^n \to B \end{array}\right\} \quad \rightsquigarrow \quad \begin{array}{l} \pi\text{OBDD } G \\ \text{für } f_1 \otimes f_2 \end{array}$$

für beliebige boolesche Funktion $\otimes : B^2 \to B$

G_1 hat Knoten $v_1, v_2, \ldots, v_{s_1}$.

G_2 hat Knoten $w_1, w_2, \ldots, w_{s_2}$.

Wir starten in Wurzeln v_1 und w_1.

Mit dem aktuellen Knotenpaar machen wir einen Synthese-Schritt.

Ergebnis des Synthese-Schritts: 1 Knoten des Ergebnis-πOBDD.

Ein Synthese-Schritt

aktuelle Knoten v_i, w_j

v_i hat Markierung $m_i \in \{x_1, x_2, \ldots, x_n, 0, 1\}$

w_j hat Markierung $m_j \in \{x_1, x_2, \ldots, x_n, 0, 1\}$

Falls $m_i \notin \{0, 1\}$ $v_{i,0}$ ist Nullnachfolger von v_i
$v_{i,1}$ ist Einsnachfolger von v_i

Falls $m_j \notin \{0, 1\}$ $w_{j,0}$ ist Nullnachfolger von w_j
$w_{j,1}$ ist Einsnachfolger von w_j

Wir unterscheiden mehrere Fälle
nach den Markierungen m_i und m_j.

1. Fall: gleiche Variable

$m_i = m_j = x_k$

Erzeuge Knoten (v_i, w_j) mit Markierung x_k.

Nullnachfolger der von Synthese-Schritt$(v_{i,0}, w_{j,0})$ erzeugte Knoten

Einsnachfolger der von Synthese-Schritt$(v_{i,1}, w_{j,1})$ erzeugte Knoten

2. Fall: verschiedene Variable

$m_i = x_k$, $m_j = x_{k'}$, x_k vor $x_{k'}$

Erzeuge Knoten (v_i, w_j) mit Markierung x_k.

Nullnachfolger — der von Synthese-Schritt$(v_{i,0}, w_j)$ erzeugte Knoten

Einsnachfolger — der von Synthese-Schritt$(v_{i,1}, w_j)$ erzeugte Knoten

3. Fall: verschiedene Variable

$m_i = x_k$, $m_j = x_{k'}$, x_k hinter $x_{k'}$

Erzeuge Knoten (v_i, w_j) mit Markierung $x_{k'}$.

Nullnachfolger — der von Synthese-Schritt$(v_i, w_{j,0})$ erzeugte Knoten

Einsnachfolger — der von Synthese-Schritt$(v_i, w_{j,1})$ erzeugte Knoten

4. Fall: eine Variable, eine Senke

$m_i = x_k,\ m_j \in \{0, 1\}$

Idee Konstante liegen in der Variablenordnung ganz hinten

Erzeuge Knoten (v_i, w_j) mit Markierung x_k.

Nullnachfolger der von Synthese-Schritt$(v_{i,0}, w_j)$ erzeugte Knoten

Einsnachfolger der von Synthese-Schritt$(v_{i,1}, w_j)$ erzeugte Knoten

5. Fall: eine Variable, eine Senke

$m_i \in \{0,1\},\ m_j = x_k$

Erzeuge Knoten (v_i, w_j) mit Markierung x_k.

Nullnachfolger — der von Synthese-Schritt$(v_i, w_{j,0})$ erzeugte Knoten

Einsnachfolger — der von Synthese-Schritt$(v_i, w_{j,1})$ erzeugte Knoten

6. Fall: zwei Senken

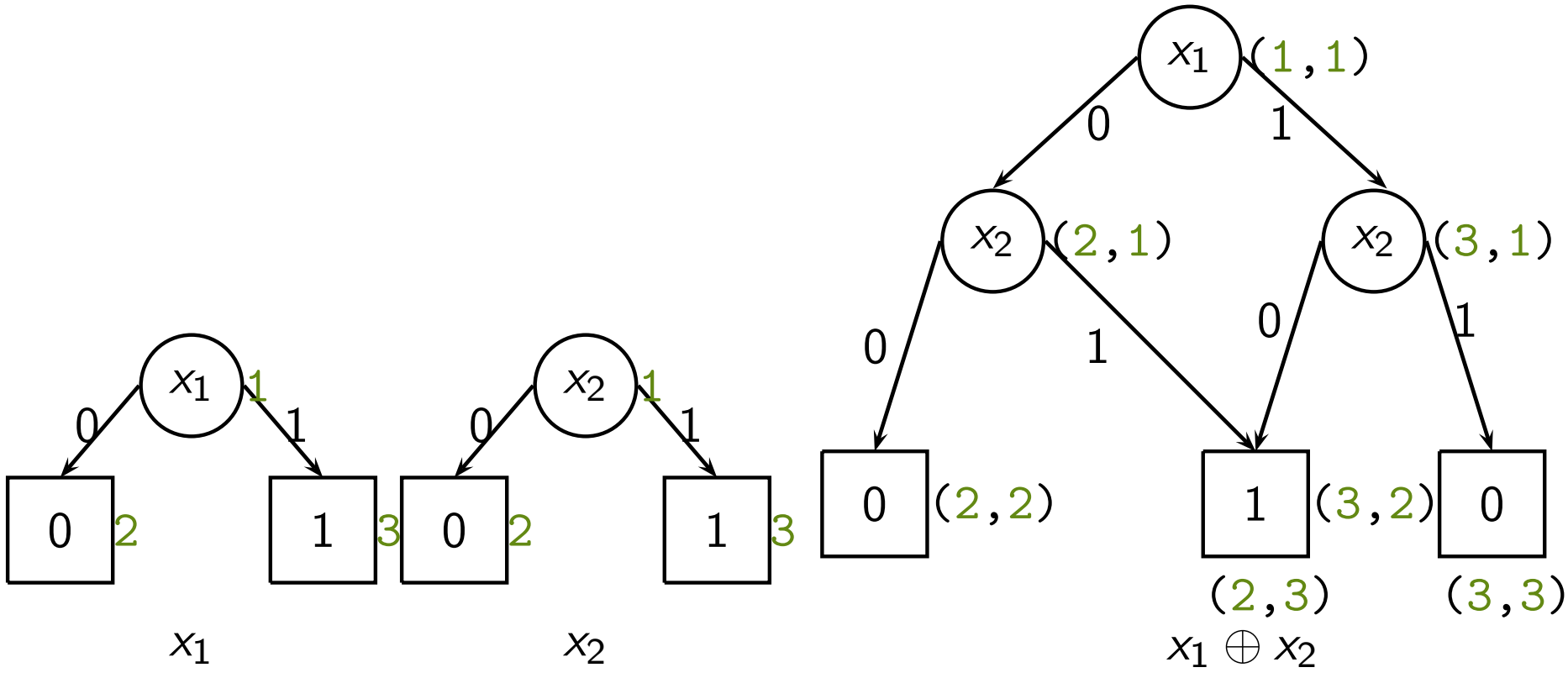
$m_i \in \{0,1\},\ m_j \in \{0,1\}$

Erzeuge mit $m_i \otimes m_j$ markierte Senke.

Zusammenfassung zum Merken

1. Erzeuge Knoten mit „weiter vorne liegender“ Markierung.
2. Wenn beide Knoten mit gleicher Variabler markiert, aus beiden Knoten fortschreiten.
3. Bei ungleicher Variablenmarkierung, nur aus „weiter vorne liegenden“ Knoten fortschreiten, im anderen Knoten warten.
4. Abbruch, wenn in beiden Knoten Senken erreicht.

Noch ein Beispiel



Variablenordnung x_1, x_2

boolesche Verknüpfung $\oplus$

Größenzuwachs bei Synthese

Wie groß wird das neue πOBDD?

Beobachtung für je zwei Knoten höchstens ein neuer Knoten

also aus πOBDDs mit s_1 und s_2 Knoten
neues πOBDD mit $\leq s_1 \cdot s_2$ Knoten

Beobachtungen

- Ergebnis-πOBDD in der Regel nicht reduziert
- Größe $s_1 \cdot s_2$ manchmal erforderlich (ohne Senken)

Operationen auf OBDDs

Erinnerung OBDDs unterstützen viele wichtige Operationen

1. Reduzierung ✓
2. Synthese ✓
3. Auswertung ✓
4. Konstantsetzung $x_i = c$ ✓
   - OBDD durchlaufen
   - Vorgänger jedes x_i-Knotens auf c-Nachfolger des x_i-Knotens umsetzen
5. Gleichheitstest ✓
   - Voraussetzung gleiche Variablenordnung π
   - beide reduzierte πOBDDs ab der Quelle parallel durchlaufen
   - bei ungleicher Markierung ungleich
   - rekursiv für Null- und Einsnachfolger

Operationen auf OBDDs (Fortsetzung)

Erinnerung OBDDs unterstützen viele wichtige Operationen

6. Null-/Einseingabe finden ✓
   - mit OBDD-Durchlauf Vorgänger-Zeiger berechnen
   - von passender Senke Aufstieg bis zur Quelle, dabei Belegung merken
7. Null-/Einseingaben zählen ✓
   - Beobachtung über Quelle laufen alle 2^n Eingaben
   - Beobachtung jede Knoten halbiert Anzahl Eingaben und leitet Hälften jeweils über 0- und 1-Nachfolger
   - OBDD von der Quelle durchlaufen
   - an jedem Knoten Anzahl Eingaben notieren
   - bei mehrfach erreichten Knoten Anzahlen addieren
   - Summe an der passenden Senke ablesen (für reduziertes πOBDD)

Schaltnetze

bis jetzt Diskussion (theoretischer) Grundlagen

Wo bleibt die Hardware?

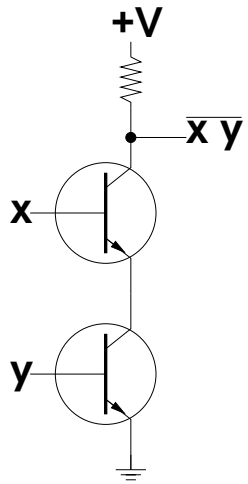
kommt jetzt aber immer noch abstrakt

Wunsch Realisierung boolescher Funktionen in Hardware

klar brauchen Realisierung einer funktional-vollständigen Menge boolescher Funktionen

Erinnerung Realisierung von NAND reicht aus

Beobachtung



realisiert NAND(x, y)

Gatter

Realisierung mit Transistoren . . . falsche Ebene!

Grundlage hier einfache logische Bausteine (Gatter)

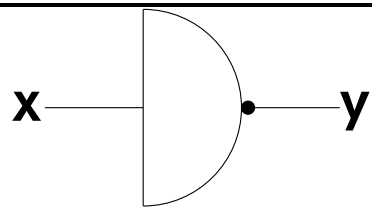
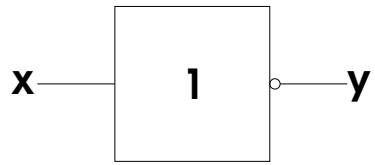
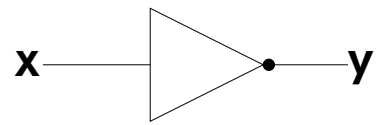
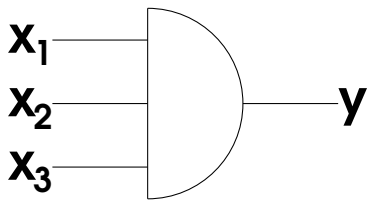
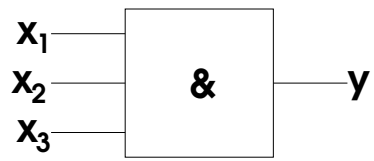
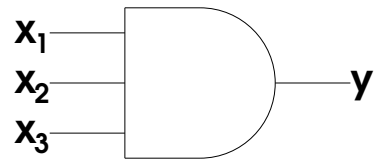
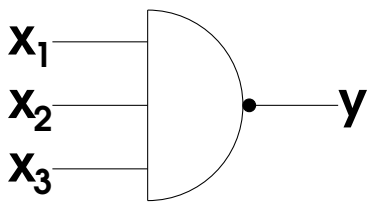
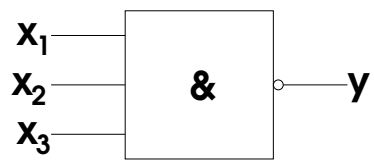
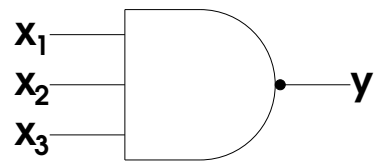
Bausteine für Negation, Konjunktion, Disjunktion, . . .

Spielregeln

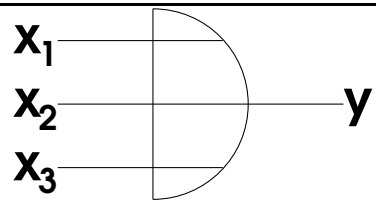
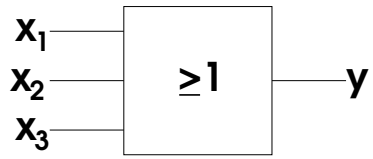
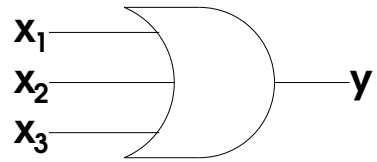
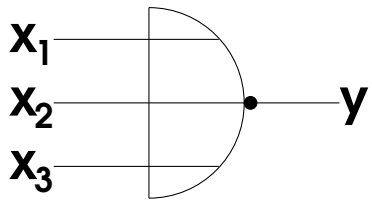
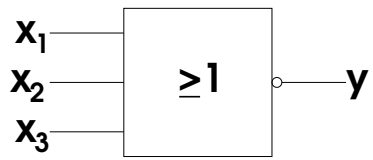
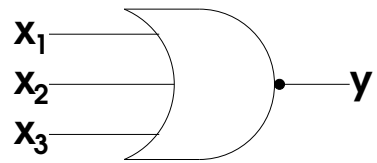
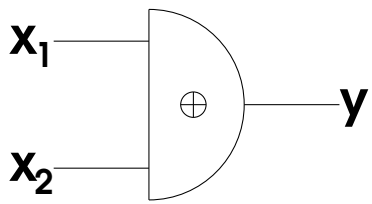
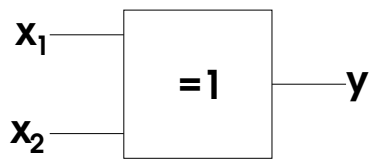
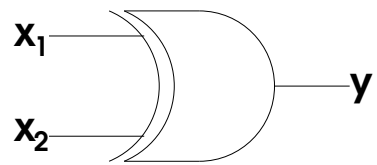
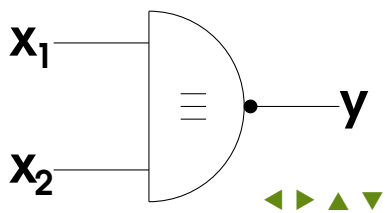
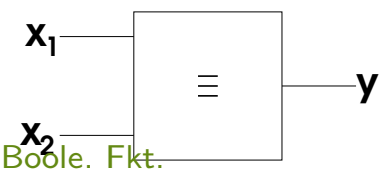
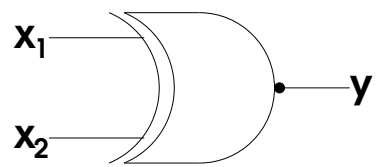
- Eingänge mit Variablen oder Konstanten belegt
- nur Verbindungen von Ausgängen zu Eingängen
- keine Kreise

Ergebnis heißt Schaltnetz

Symbole für Gatter (1)

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

Symbole für Gatter (2)

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

Schaltnetz-Bewertung

Und jetzt beliebige Schaltnetze entwerfen?

mindestens relevant Größe und Geschwindigkeit

- Schaltnetzgröße (= Anzahl der Gatter) wegen Kosten, Stromverbrauch, Verlustleistung, Zuverlässigkeit, ...
- Schaltnetztiefe (= Länge längster Weg Eingang $\rightsquigarrow$ 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)

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 □

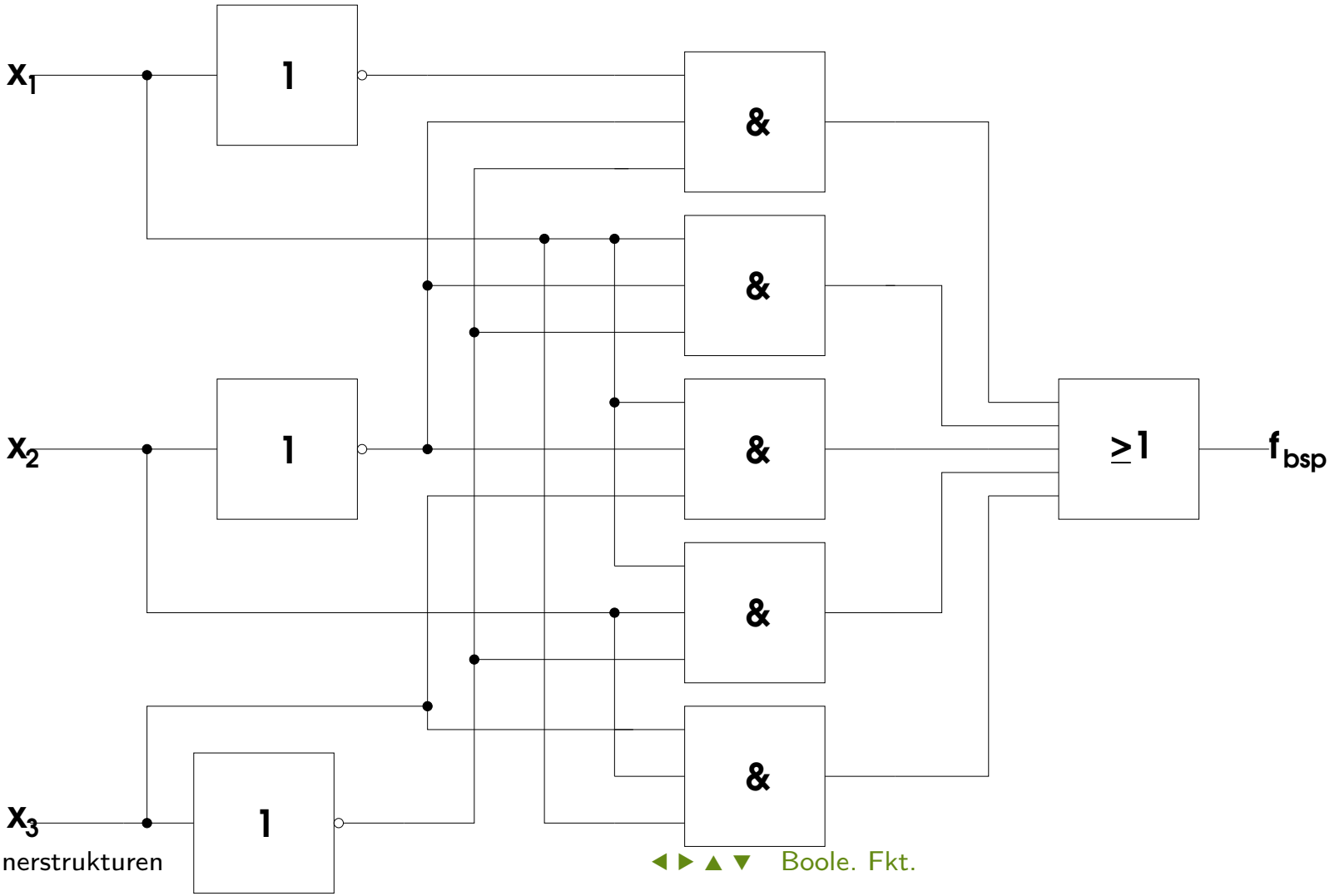
Probleme

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

Beispiel: DNF

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

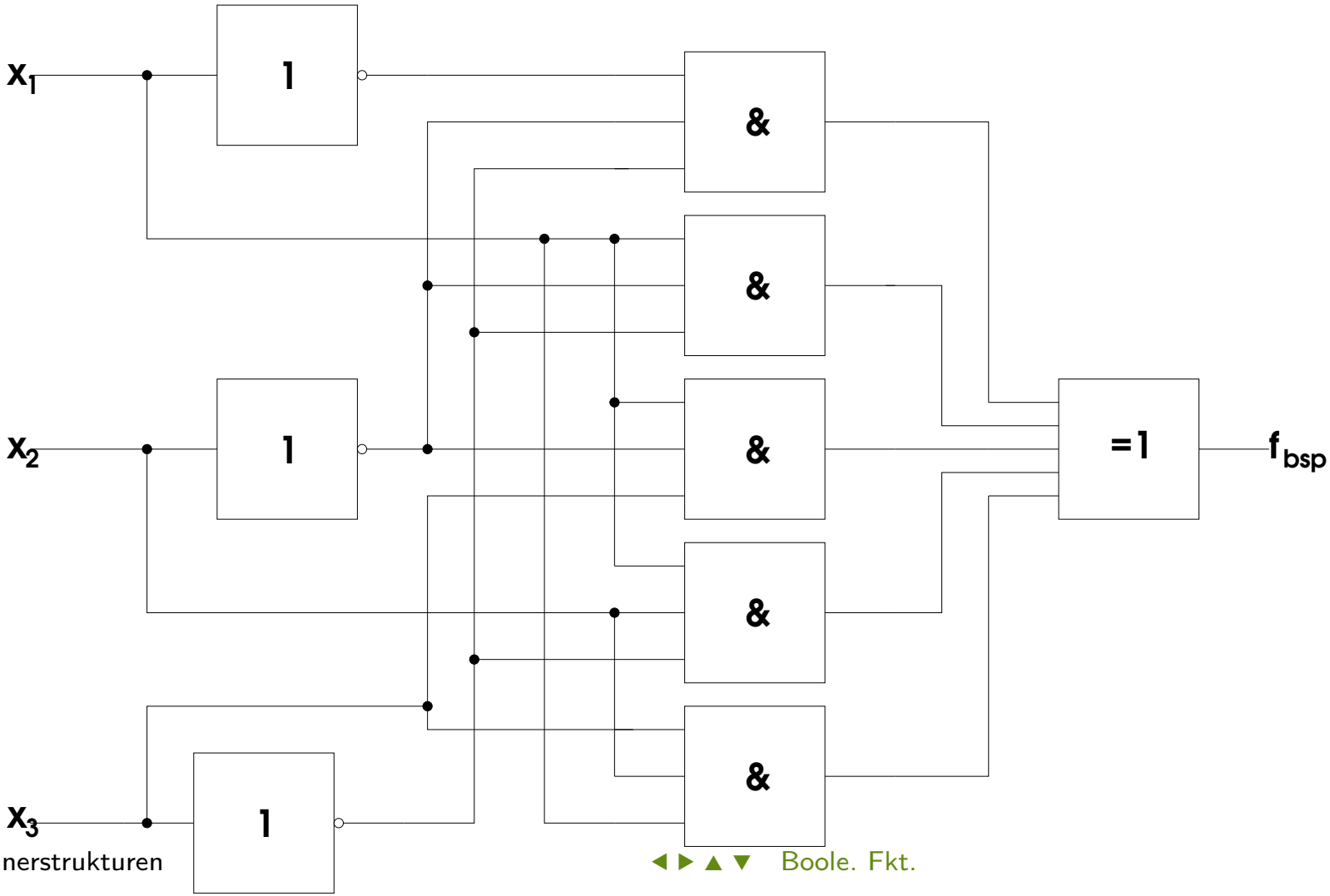
$f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_2}\,\overline{x_3} \vee x_1\,\overline{x_2}\,x_3 \vee x_1\,x_2\,\overline{x_3} \vee x_1\,x_2\,x_3$



Beispiel: RNF

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

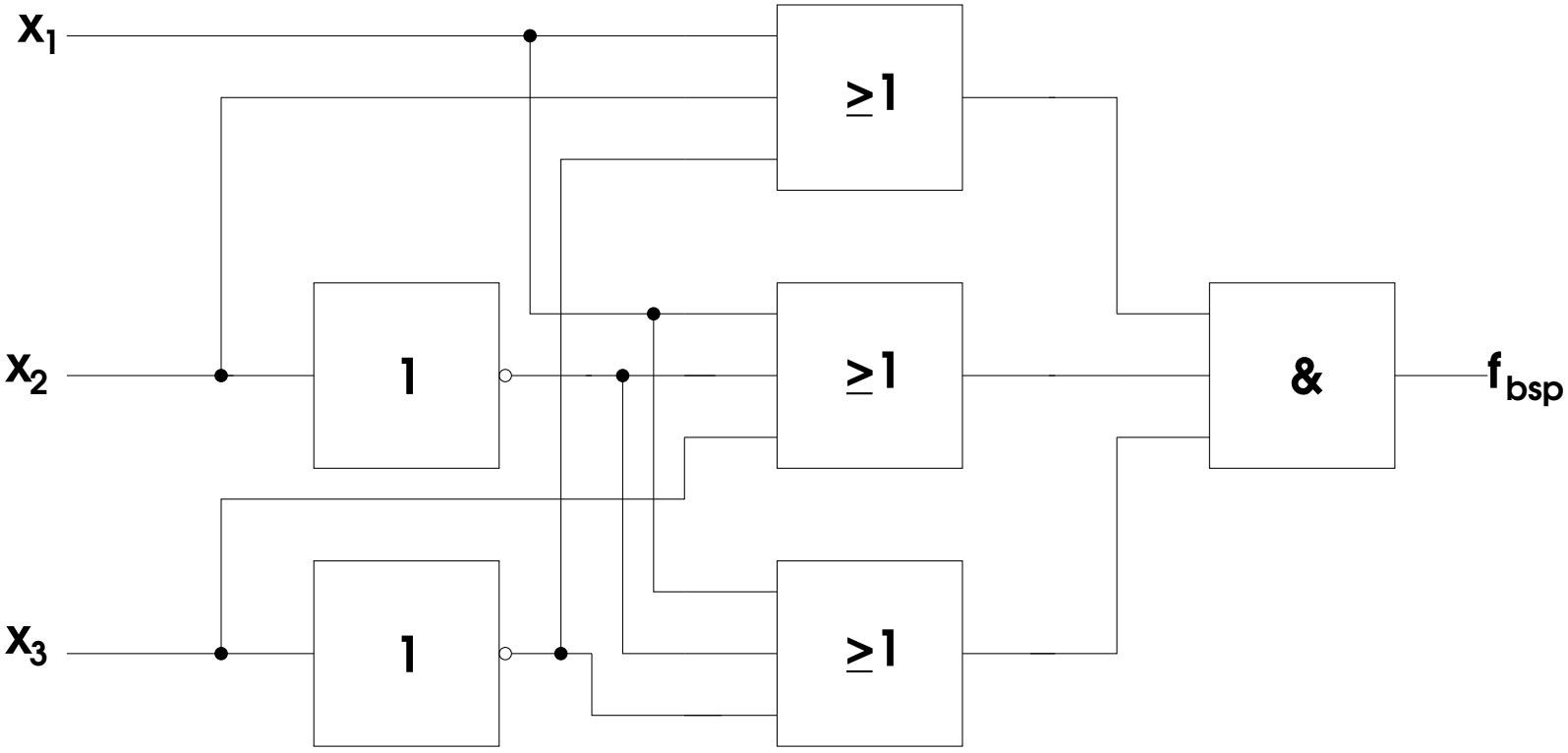
$f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} \oplus x_1\,\overline{x_2}\,\overline{x_3} \oplus x_1\,\overline{x_2}\,x_3 \oplus x_1\,x_2\,\overline{x_3} \oplus x_1\,x_2\,x_3$



Beispiel: KNF

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

$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3})$



Multiplexer

eine weitere Beispielfunktion ...

aber eine in der Praxis sehr wichtige diesmal!

$\mathrm{MUX}_d\left(y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}\right) = x_{(y_1\, y_2 \cdots y_d)_2}$

Beispiel

y_1	y_2	y_3	$\mathrm{MUX}_3(y_1, y_2, y_3, x_0, x_1, \ldots, x_7)$
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

Multiplexer

Warum ist MUX in der Praxis wichtig?

direkte Speicheradressierung

OBDD-Realisierung

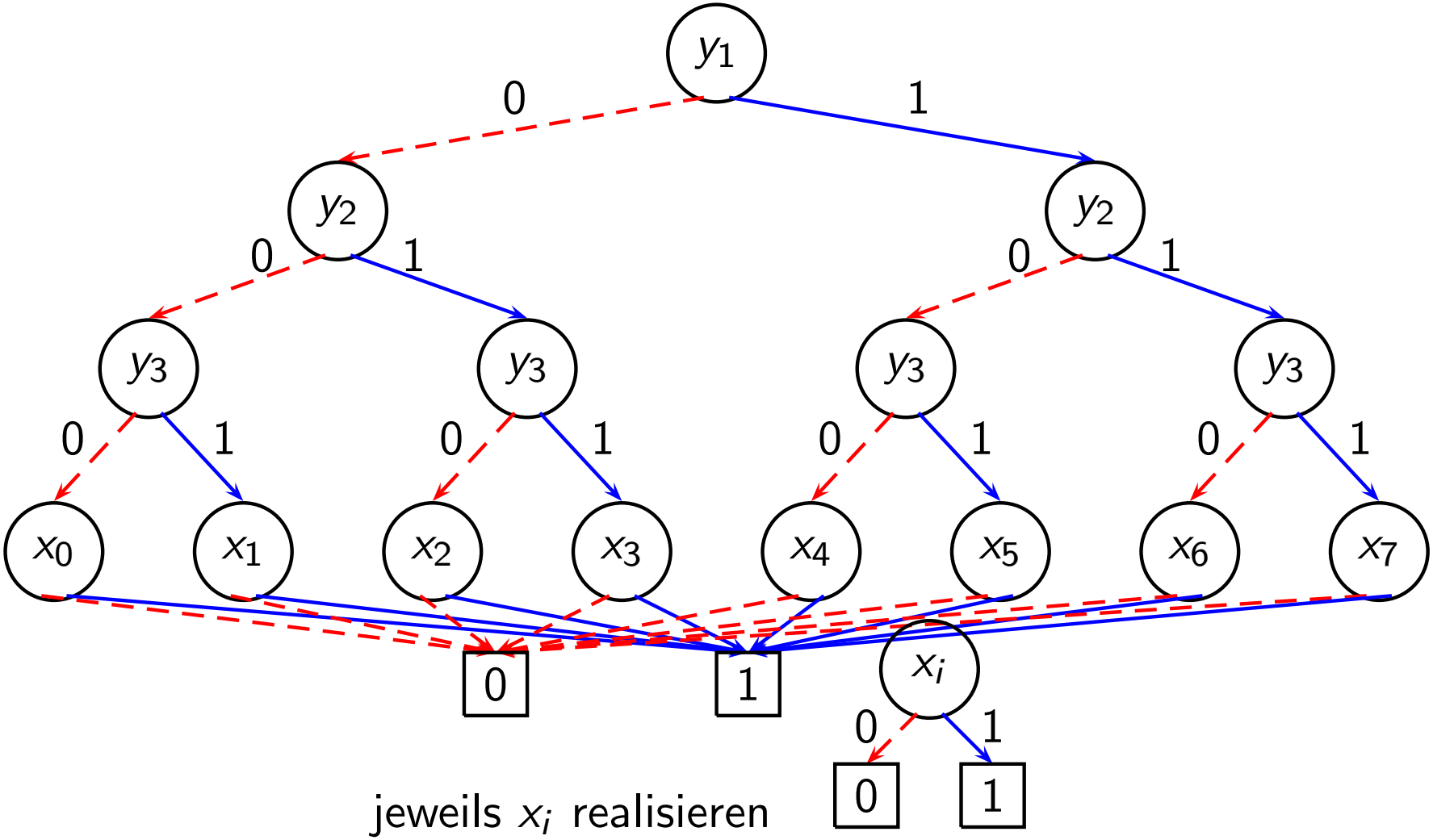
Welche Variablenordnung π?

wohl sinnvoll Adressvariablen zuerst

denn Wir müssen uns sonst alle Datenvariablen merken!

OBDD-Realisierung $\mathbf{MUX_3}$

Variablenordnung $\pi = (y_1, y_2, y_3, x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$



jeweils x_i realisieren

Einfluss von π auf OBDD-Größe

Beispiel MUX_d

gesehen Variablenordnung $\pi = (y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1})$

Größe des reduzierten πOBDDs?

oben vollständiger Binärbaum über $y_1, y_2, \ldots, y_d$
$2^0 + 2^1 + \cdots + 2^{d-1} = 2^d - 1$ Knoten

unten je ein x_i-Knoten und zwei Senken
$2^d + 2$ Knoten

zusammen $2^d - 1 + 2^d + 2 = 2^{d+1} + 1$ Knoten

Einfluss von π auf OBDD-Größe

Beispiel MUX_d

gesehen Variablenordnung $\pi = (y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1})$
Größe des reduzierten πOBDD $2^{d+1} + 1$

Betrachte Variablenordnung $\pi = (x_0, x_1, \ldots, x_{2^d-1}, y_1, y_2, \ldots, y_d)$

Größe des reduzierten πOBDDs?

Behauptung $\forall x \neq x' \in \{0,1\}^{2^d}$: nach Lesen von x bzw. x' verschiedene Knoten erreicht

Beweis durch Widerspruch

Widerspruchsbeweis zur OBDD-Größe

Behauptung $\forall x \neq x' \in \{0,1\}^{2^d}$: nach Lesen von x bzw. x' verschiedene Knoten erreicht

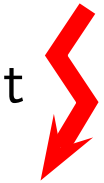
Annahme für $x \neq x' \in \{0,1\}^{2^d}$ gleiche Knoten v erreicht

Betrachte $i = \min\{j \mid x_j \neq x'_j\}$

Betrachte $y_1, y_2, \ldots, y_d$ mit $(y_1 y_2 \cdots y_d)_2 = i$

Beobachtung $\mathrm{MUX}_d(y_1, y_2, \ldots, y_d, x) = x_i$
$\neq x'_i = \mathrm{MUX}_d(y_1, y_2, \ldots, y_d, x')$

aber OBDD berechnet gleichen Wert, da gleicher Knoten erreicht



also minimales OBDD für $\pi = (x_0, x_1, \ldots, x_{2^d-1}, y_1, y_2, \ldots, y_d)$ hat Größe mindestens $2^{2^d} - 1$ (Binärbaum d. Tiefe 2^d)

d	min. OBDD-Größe $\pi = (d, x)$	min. OBDD-Größe $\pi = (x, d)$
2	9	≥ 15
4	33	$\geq 65\,535$
8	513	$\geq 1{,}1579 \cdot 10^{77}$

Schaltnetz für MUX_3