Kapitel 7

Formale Spezifikation von Hardware:
1. Boolesche Ausdrücke
2. **Binäre Entscheidungsdiagramme (BDDs)**
3. Anwendung: Formale Verifikation

Albert-Ludwigs-Universität Freiburg

Dr. Tobias Schubert, Dr. Ralf Wimmer

Professur für Rechnerarchitektur

WS 2016/17

Motivation: Eindeutigkeit der Darstellung boolescher Funktionen

- Die Darstellung einer gegebenen booleschen Funktion als Schaltkreis oder als boolescher Ausdruck ist im Allgemeinen nicht eindeutig.

- **Beispiel**: „$x_1 \cdot \sim x_2 + x_3$" und „$x_1 \cdot \sim x_2 + x_3 + x_3$".

- Wir kennen eindeutige Darstellungen einer booleschen Funktion.

- **Zum Beispiel**:
  - Funktionstabelle
  - Kanonische disjunktive Normalform (alle Minterme)

- **Aber**: Diese eindeutigen Darstellungen sind „sehr sperrig".
  $\Rightarrow$ 2^n Einträge/alle Minterme für Funktionen in n Variablen.

- **Ziel**: Finde eine eindeutige, aber kompakte Darstellung boolescher Funktionen.

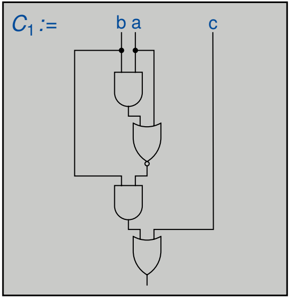
Wozu Eindeutigkeit?

- Zur schnellen Überprüfung, ob zwei boolesche Funktionen äquivalent sind.
- **Beispiel**:
  - Ein Designer hat Optimierungen an der Hardware vorgenommen und 15 Gatter gespart. Hat er dabei vielleicht die Funktionalität verändert?

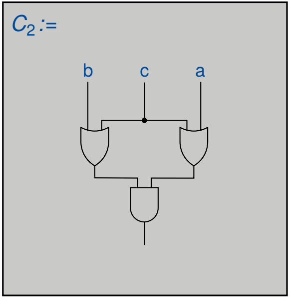
  $\rightarrow$ Erzeuge eindeutige Darstellung für alten und neuen Block und vergleiche.

$\rightarrow$ Dieses Vorgehen nennt man formalen Äquivalenzcheck.

Illustration: Formaler Äquivalenztest



$\stackrel{?}{=}$



Eindeutige Darstellung der durch C_1 realisierten booleschen Funktion.

$\stackrel{?}{=}$

Eindeutige Darstellung der durch C_2 realisierten booleschen Funktion.

Binäre Entscheidungsdiagramme (Binary Decision Diagrams)

- BDDs repräsentieren boolesche Funktionen eindeutig und (oft) kompakt zugleich.
  - → Enthält genau die gleiche Information, wie eine Funktionstabelle.
- Es gibt Verfahren, um schnell ein BDD aus einem Schaltkreis zu erzeugen.
  - → Man muss nicht erst die Funktionstabelle erzeugen.
- Es existieren Software-Packages, die Erzeugung und Manipulation von BDDs unterstützen.
  - CUDD von der Colorado University: `http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html`
  - Zahlreiche Optimierungen bei Cache-Ausnutzung, Garbage Collection, usw.

Binary Decision Diagram – Syntax

- Sei $X_n = \{x_1, \ldots, x_n\}$ eine endliche Menge von Variablen.

Definition

Ein BDD ist

- ein azyklischer gerichteter Graph $G = (V, E)$ mit
  - genau einer Wurzel,
  - $outdeg(v) = 2$ für alle inneren Knoten v.
- Jeder Knoten v hat eine Markierung
  - $label(v) = \begin{cases} x_i \in X_n, & v \text{ ist innerer Knoten} \\ c \in \{0,1\}, & v \text{ ist Blatt} \end{cases}$
- Die Nachfolger (Kinder) eines inneren Knoten v sind
  - das Low-Kind: $low(v)$,
  - das High-Kind: $high(v)$.

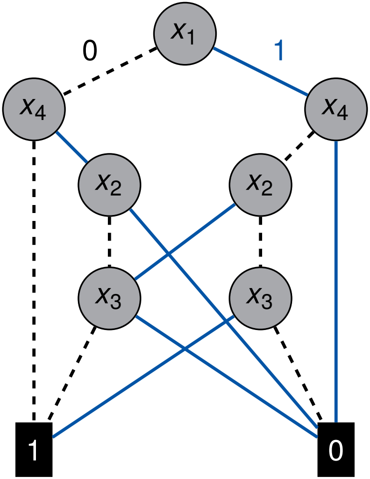
BDDs: Semantik

- Gegeben sei ein BDD.

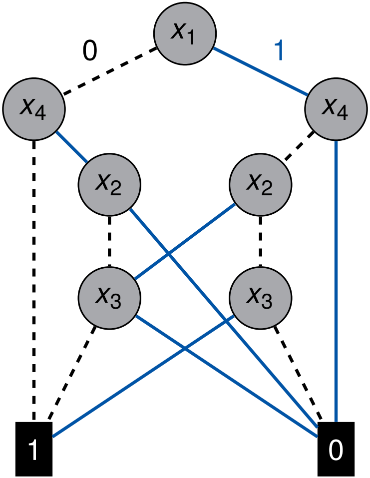
Definition

- Sei f_v die Funktion, die ein Teil-BDD ausgehend von Knoten v beschreibt.
  - Ist v ein Blatt, so beschreibt v die konstante Funktion, die jede Variablenbelegung auf $label(v) \in \{0,1\}$ abbildet.
  - Ist v ein innerer Knoten mit $label(v) = x_j$, so gilt die Kompositionsregel: $f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$.

BDDs: Beispiel

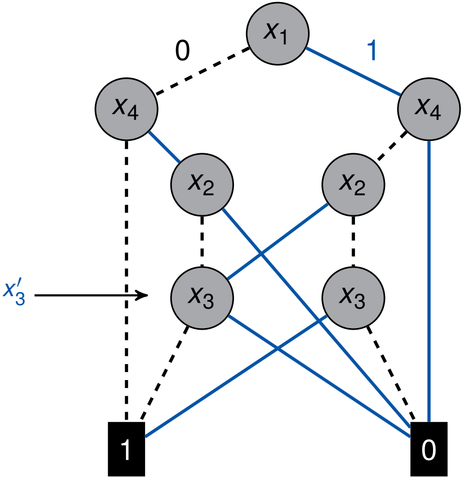


BDDs: Beispiel



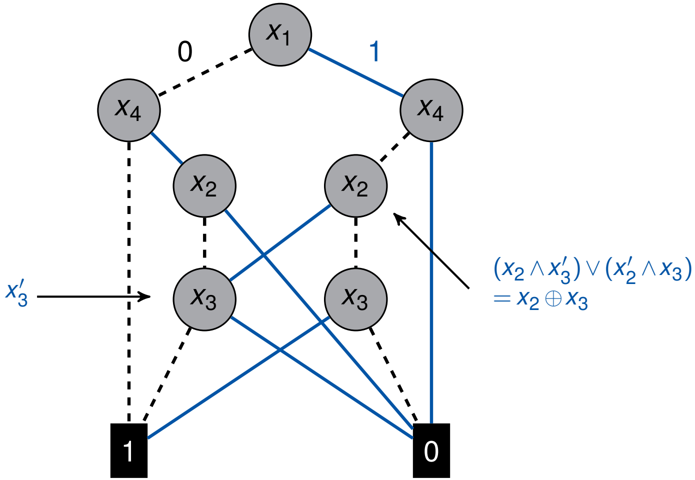
$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

BDDs: Beispiel



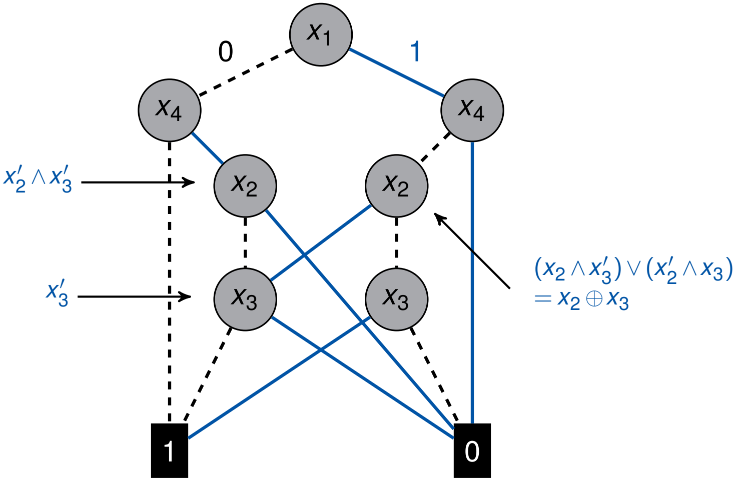
$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

BDDs: Beispiel



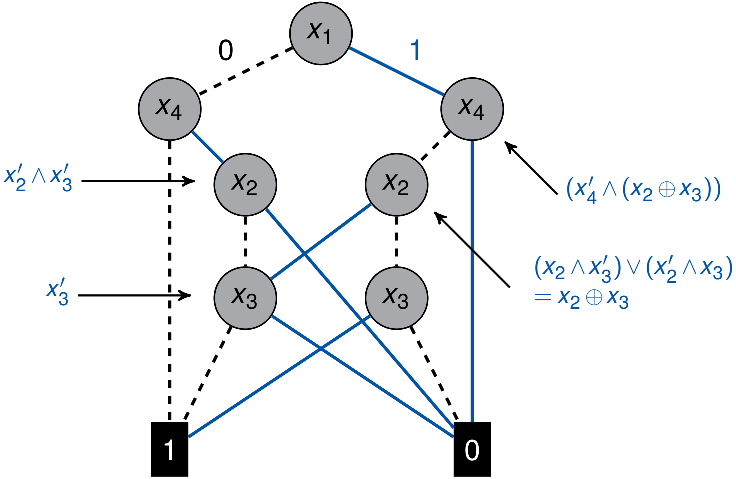
$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

BDDs: Beispiel



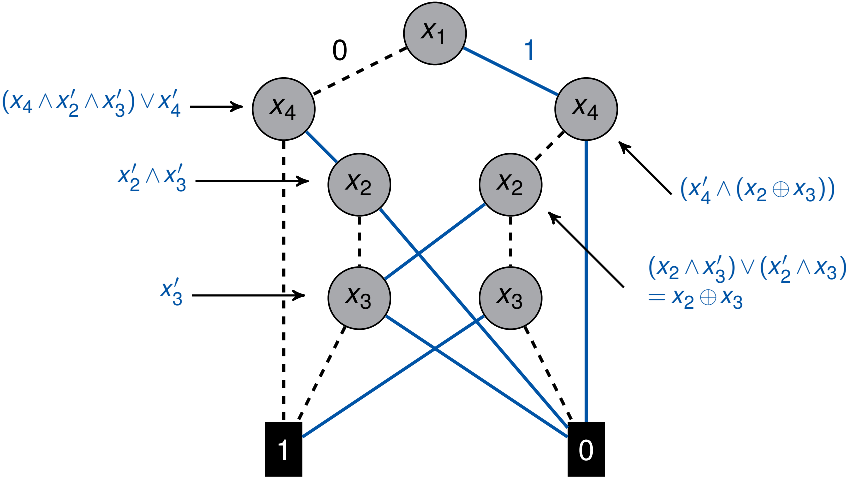
$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

BDDs: Beispiel



$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

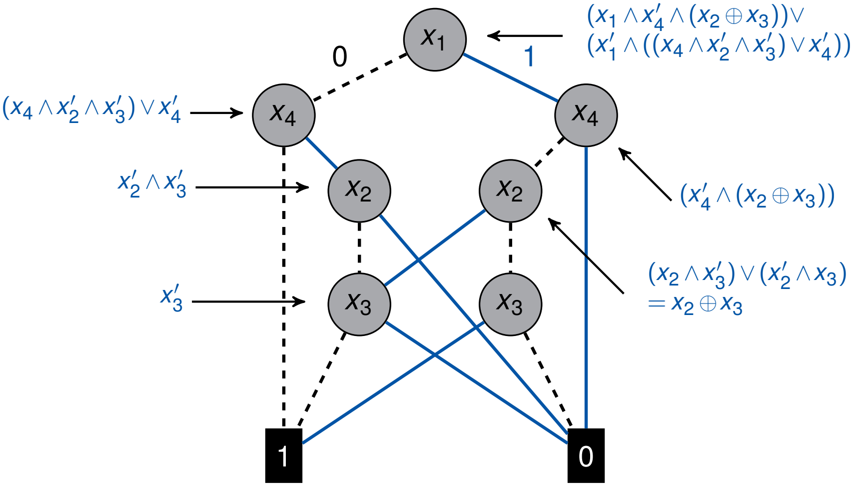
BDDs: Beispiel



$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

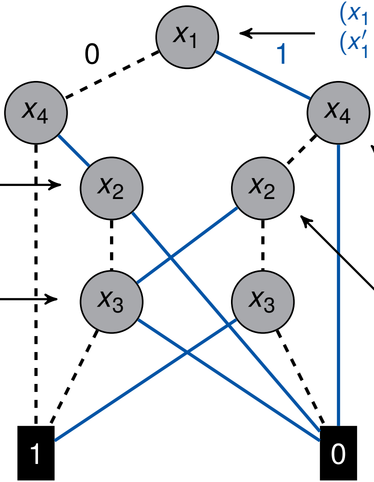
UNI FREIBURG

BDDs: Beispiel



$$f_v = (x_j \wedge f_{high(v)}) \vee (x_j' \wedge f_{low(v)})$$

BDDs: Beispiel



$(x_1 \wedge x'_4 \wedge (x_2 \oplus x_3)) \vee$
$(x'_1 \wedge ((x_4 \wedge x'_2 \wedge x'_3) \vee x'_4))$

$(x_4 \wedge x'_2 \wedge x'_3) \vee x'_4$

$x'_2 \wedge x'_3$

x'_3

$(x'_4 \wedge (x_2 \oplus x_3))$

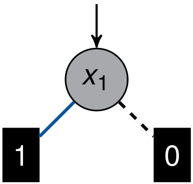
$(x_2 \wedge x'_3) \vee (x'_2 \wedge x_3)$
$= x_2 \oplus x_3$

$$f_v = (x_j \wedge f_{high(v)}) \vee (x'_j \wedge f_{low(v)})$$

BDDs von typischen Funktionen

Seien x_1, x_2 beliebige boolsche Variablen.

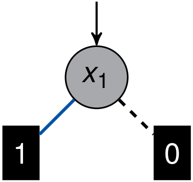
x_1:



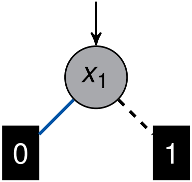
BDDs von typischen Funktionen

Seien x_1, x_2 beliebige boolsche Variablen.

x_1:



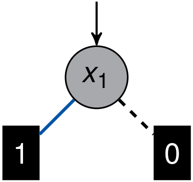
x_1':



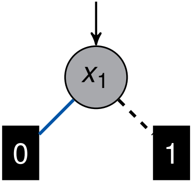
BDDs von typischen Funktionen

Seien x_1, x_2 beliebige boolsche Variablen.

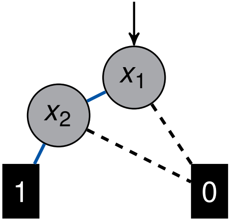
x_1:



x_1':



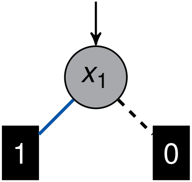
$x_1 \cdot x_2$:



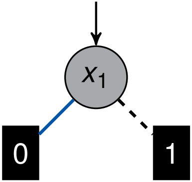
BDDs von typischen Funktionen

Seien x_1, x_2 beliebige boolsche Variablen.

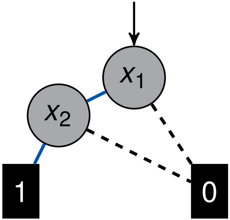
x_1:



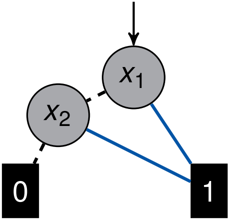
x_1':



$x_1 \cdot x_2$:



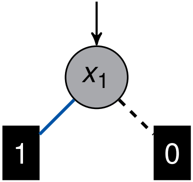
$x_1 + x_2$:



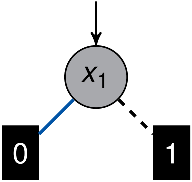
BDDs von typischen Funktionen

Seien x_1, x_2 beliebige boolsche Variablen.

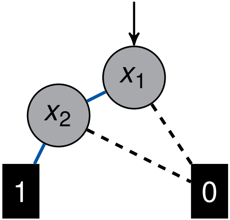
x_1:



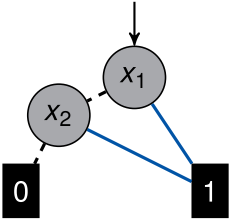
x_1':



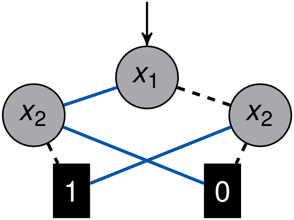
$x_1 \cdot x_2$:



$x_1 + x_2$:



$x_1 \oplus x_2$:



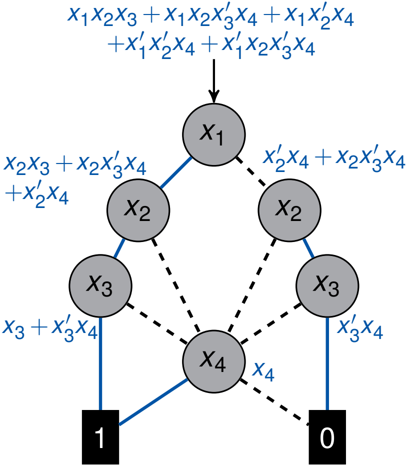
Variablenordnung

Definition

- Ein BDD heißt geordnet, wenn auf jedem Pfad von der Wurzel zu einem Blatt jede Variable höchstens einmal (oder keinmal) als Markierung vorkommt und die Variablen stets in der selben Reihenfolge abgefragt werden.

- Diese Reihenfolge heißt Variablenordnung.

Beispiel für unterschiedliche Variablenordnung

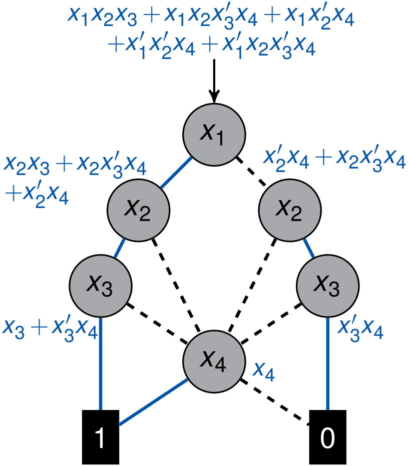


Variablenordnung

x_1, x_2, x_3, x_4

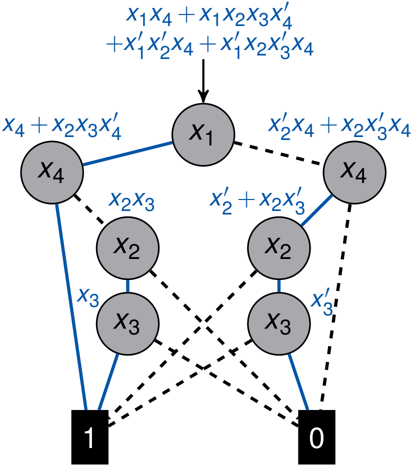
UNI FREIBURG

Beispiel für unterschiedliche Variablenordnung



Variablenordnung

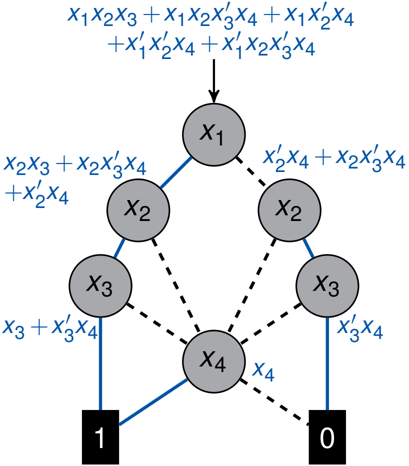
x_1, x_2, x_3, x_4



Variablenordnung

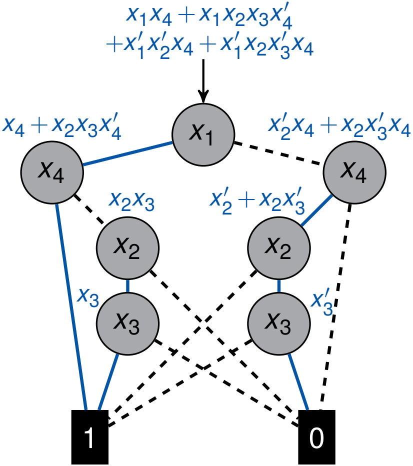
x_1, x_4, x_2, x_3

Beispiel für unterschiedliche Variablenordnung



Variablenordnung

x_1, x_2, x_3, x_4

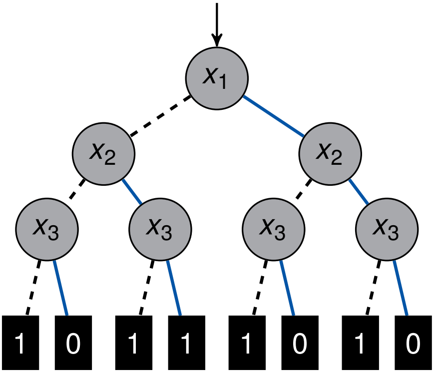


Variablenordnung

x_1, x_4, x_2, x_3

$\Rightarrow$ Beide BDDs beschreiben die boolesche Funktion $x_1x_2x_3 + x_2'x_4 + x_3'x_4$.

Vollständiges BDD



x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

- Ist ein geordnetes BDD, wobei alle Pfade von der Wurzel zu einem Blatt maximale Länge haben und $indeg(v) = 1$ für alle Knoten v.
- Entspricht genau der Funktionstabelle (Beweis klar)

Zusammenhang: BDD $\leftrightarrow$ Boolesche Funktion

Lemma 1

Jedes (geordnete) BDD stellt eine boolesche Funktion dar.

- **Beweis**: Ist v die Wurzel des BDDs, dann erhält man mit der Kompositionsregel die dargestellte boolesche Funktion f_v.

Lemma 2

Zu jeder booleschen Funktion existiert ein (geordnetes) BDD, das diese darstellt.

- **Beweis**: Konstruiere vollständiges BDD aus Funktionstabelle.

Eindeutigkeit von BDDs

- Nicht geordnete BDDs sind nicht eindeutige Darstellungen, ebenso BDDs mit unterschiedlicher (nicht fester) Variablenordnung.
- Die vollständigen BDDs sind eindeutige Darstellungen (da die Funktionstabelle eindeutig ist).
- **Aber**: Ineffizient $\rightarrow$ Reduktion von BDDs

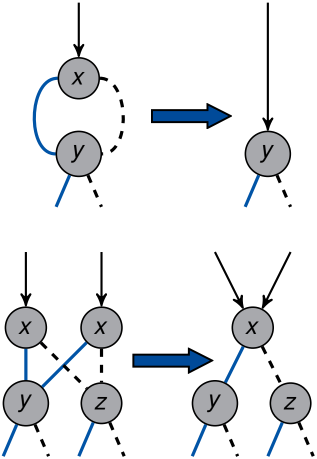
Reduzierte BDDs

Idee: Stelle Information möglichst kompakt dar, „isomorphe Teil-BDDs" werden identifiziert.

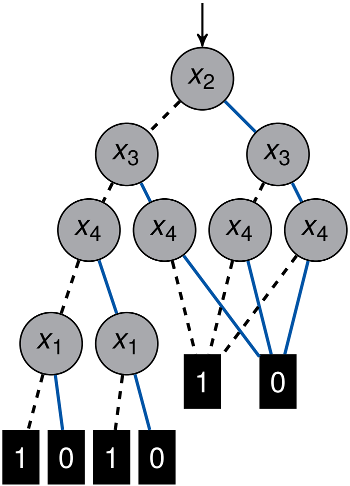
Definition

Ein BDD heißt reduziert, wenn es

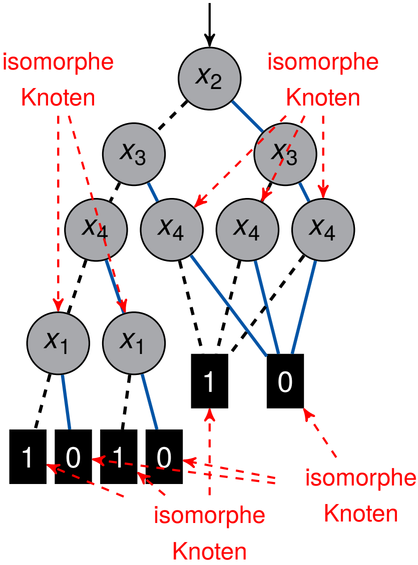
- keinen nichtterminalen Knoten v gibt, mit Label x und $f_{high(v)} = f_{low(v)}$.
- keine verschiedenen Knoten v und w gibt, die gleich markiert sind und deren low- und high-Kind (falls existent) jeweils identisch sind.



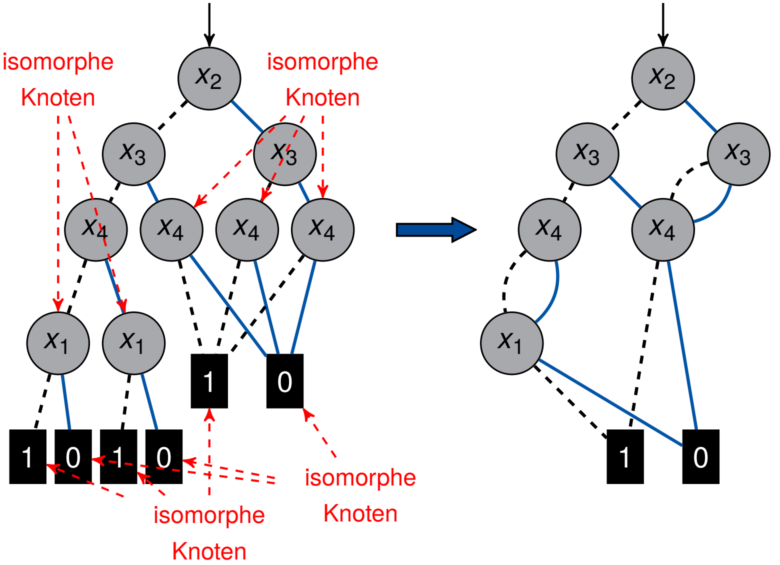
Beispiel für reduzierte und nicht reduzierte BDDs (1/2)



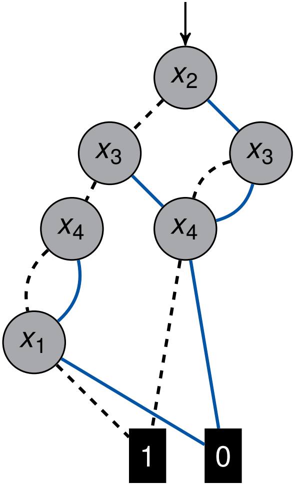
Beispiel für reduzierte und nicht reduzierte BDDs (1/2)



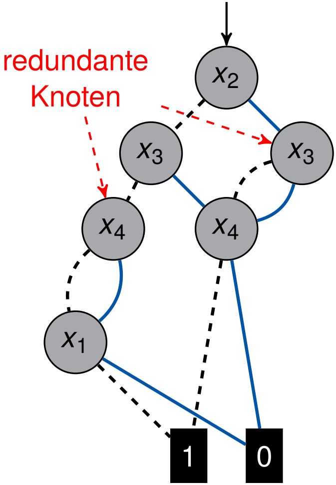
Beispiel für reduzierte und nicht reduzierte BDDs (1/2)



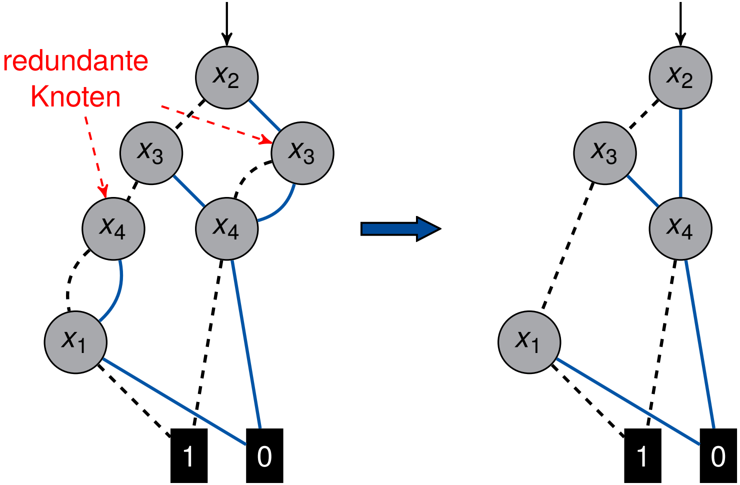
Beispiel für reduzierte und nicht reduzierte BDDs (2/2)



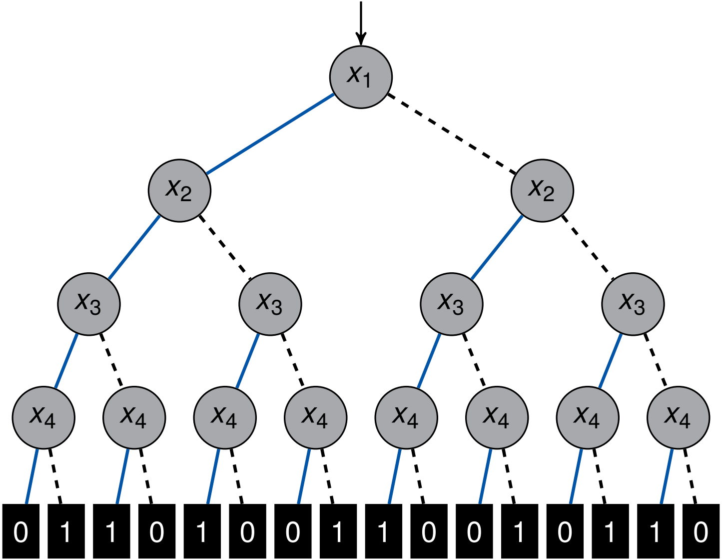
Beispiel für reduzierte und nicht reduzierte BDDs (2/2)



Beispiel für reduzierte und nicht reduzierte BDDs (2/2)

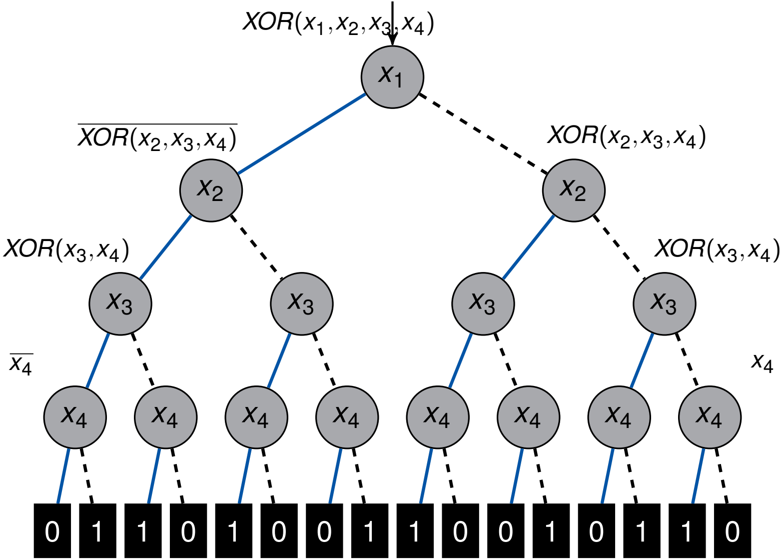


Beispiel: Reduktion eines BDDs (1/7)

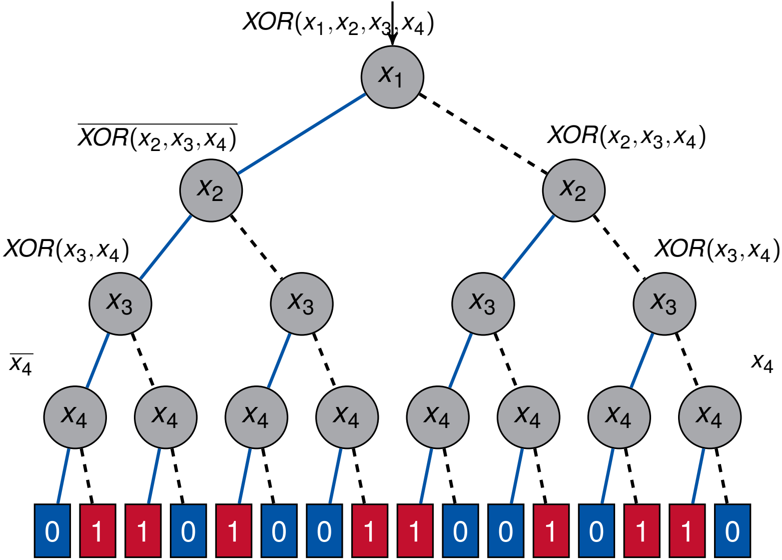


UNI FREIBURG

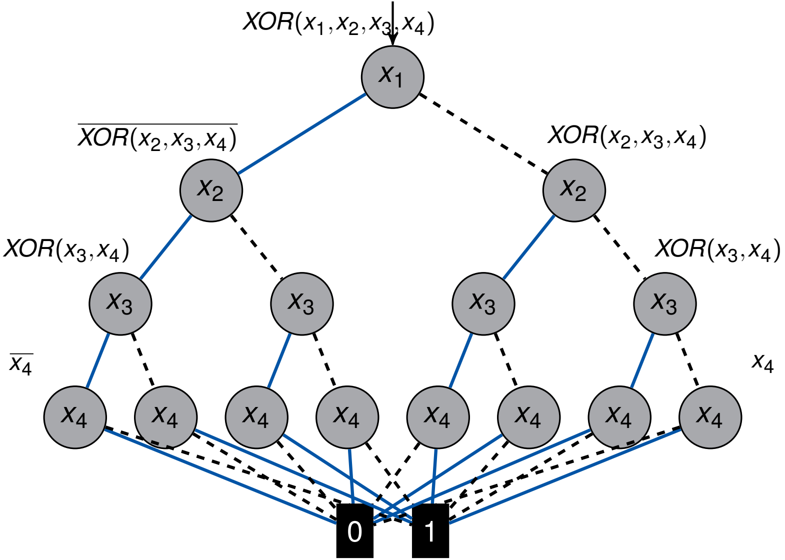
Beispiel: Reduktion eines BDDs (1/7)



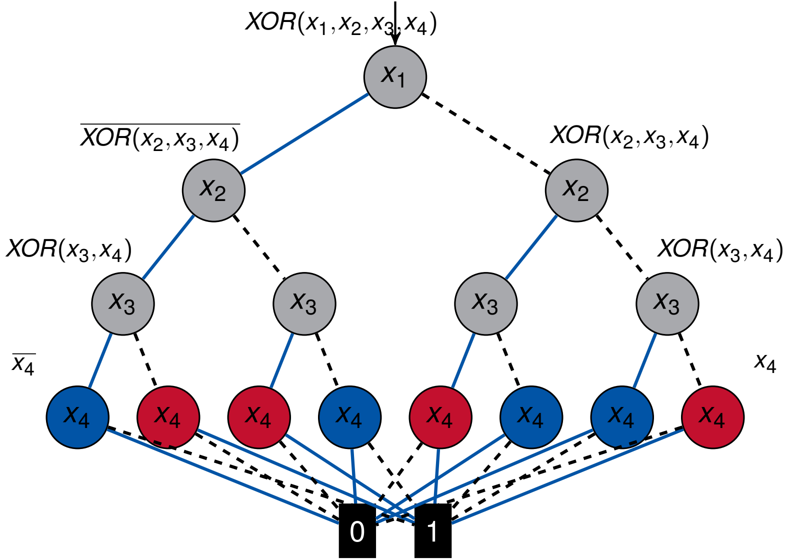
Beispiel: Reduktion eines BDDs (2/7)



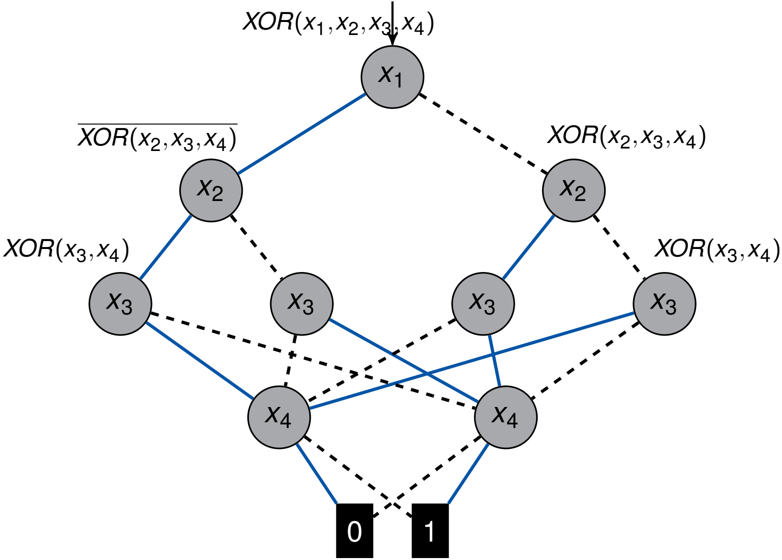
Beispiel: Reduktion eines BDDs (3/7)



Beispiel: Reduktion eines BDDs (4/7)

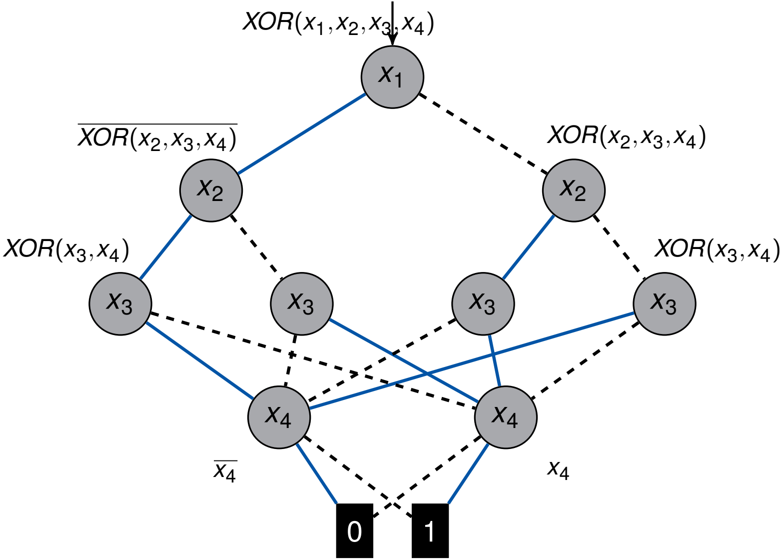


Beispiel: Reduktion eines BDDs (5/7)

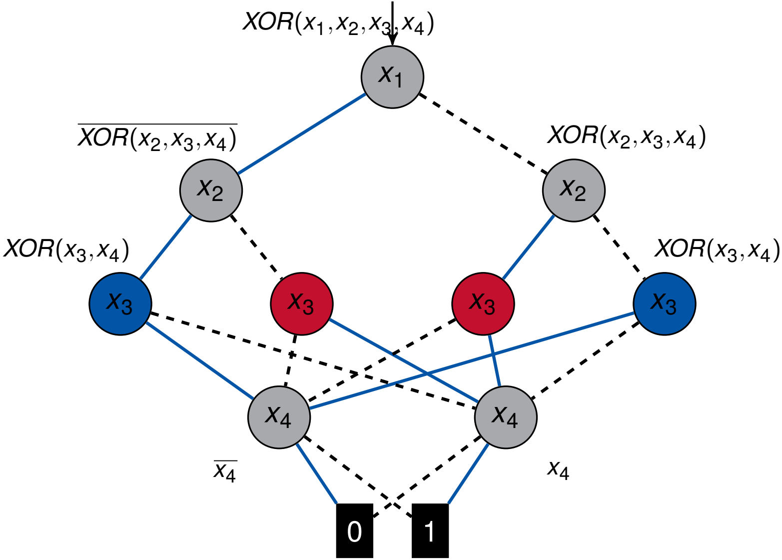


UNI FREIBURG

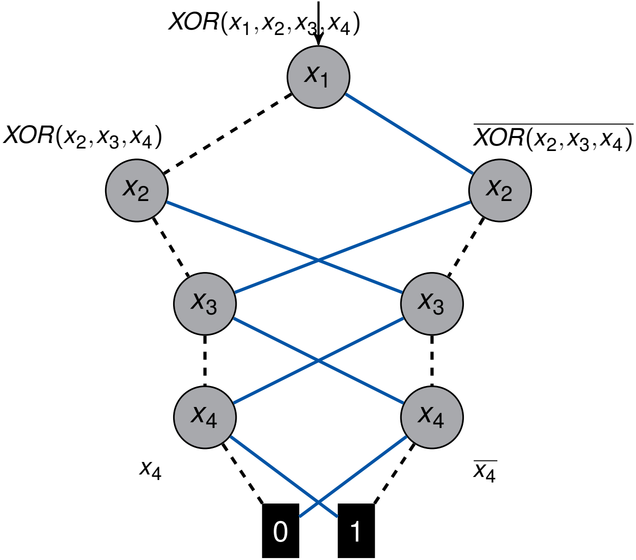
Beispiel: Reduktion eines BDDs (5/7)



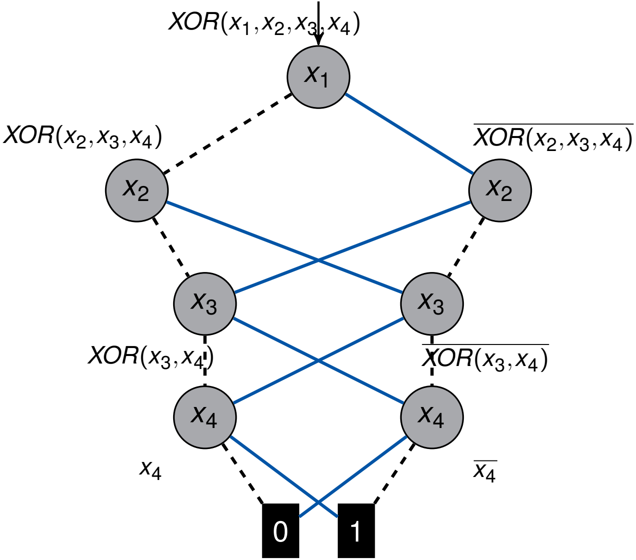
Beispiel: Reduktion eines BDDs (6/7)



Beispiel: Reduktion eines BDDs (7/7)



Beispiel: Reduktion eines BDDs (7/7)



Satz von Bryant (1986)

Satz

ROBDDs (Reduced Ordered BDDs, geordnete, reduzierte BDDs) sind kanonische Darstellungen boolescher Funktionen.

- Ohne Beweis.
- **Bedeutung**: Für eine fixe Variablenordnung ist ein ROBDD eine eindeutige Darstellung für eine boolesche Funktion.
- **Notation**: Ab sofort bezeichen wir ROBDDs als BDDs.

UNI FREIBURG

Kofaktoren und Shannon-Zerlegung

- Für eine boolesche Funktion $f : \mathbb{B}^n \to \mathbb{B}$ heißt $f_{x_i=1} : \mathbb{B}^n \to \mathbb{B}$ definiert durch $f_{x_i=1}(\alpha_1, \dots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \dots, \alpha_n) = f(\alpha_1, \dots, \alpha_{i-1}, 1, \alpha_{i+1}, \dots, \alpha_n)$ für alle $\alpha \in \{0,1\}^n$ der Kofaktor von f nach $x_i = 1$.

- Entsprechend heißt $f_{x_i=0} = f(\alpha_1, \dots, \alpha_{i-1}, 0, \alpha_{i+1}, \dots, \alpha_n)$ der Kofaktor von f nach $x_i = 0$.

Shannonscher Entwicklungssatz

Es gilt für beliebiges f:

$$f = (x_i \wedge f_{x_1=1}) \vee (x_i' \wedge f_{x_i=0})$$

- $f = (x_i \wedge f_{x_i=1}) \vee (x_i' \wedge f_{x_i=0})$ heißt Shannon-Zerlegung von f.

- Sei v ein innerer Knoten eines geordneten BDDs, der mit x_i markiert ist.
- Dann kommt x_i in Kindern von v nicht vor (sonst mehrfaches Vorkommen auf dem gleichen Pfad).
- Also sind die Subfunktionen $f_{high(v)}$, $f_{low(v)}$ von f_v von x_i unabhängig, d.h. für alle $\alpha \in \{0,1\}^n$ gilt:
  - $f_{high(v)}(\alpha_1, \dots, \alpha_{i-1}, 1, \alpha_{i+1}, \dots, \alpha_n) = f_{high(v)}(\alpha_1, \dots, \alpha_{i-1}, 0, \alpha_{i+1}, \dots, \alpha_n)$
  - $f_{low(v)}(\alpha_1, \dots, \alpha_{i-1}, 1, \alpha_{i+1}, \dots, \alpha_n) = f_{low(v)}(\alpha_1, \dots, \alpha_{i-1}, 0, \alpha_{i+1}, \dots, \alpha_n)$
- Es folgt dann: $f_{high(v)} = f_{v,x_i=1}, \quad f_{low(v)} = f_{v,x_i=0}$.
- Somit wird an v die Shannon-Zerlegung berechnet: $f_v = (x_i \wedge f_{high(v)}) \vee (x_i' \wedge f_{low(v)}) = (x_i \wedge f_{v,x_i=1}) \wedge (x_i' \wedge f_{v,x_i=0})$.

Konstruktion von BDDs

- Über die Kofaktoren.
  - **Beispiel**: Um ein bdd_f von $f = x_1x_2x_3 + x_2'x_4 + x_3'x_4$ bzgl. der Variablenordnung x_1, x_2, x_3, x_4 zu konstruieren, konstruiere BDDs von $f_{x_1=1} = x_2x_3 + x_2'x_4 + x_3'x_4$ und $f_{x_1=0} = x_2'x_4 + x_3'x_4$. Sie stellen die Söhne des Knotens x_1 dar. Wende dies rekursiv an.

- Aus einem booleschen Ausdruck über boolesche Operationen.

- Aus einem Schaltkreis über boolesche Operationen (Symbolische Simulation).

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

b	c	$f_{a=1}$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

b	c	$f_{a=1}$
0	0	0
0	1	1
1	0	1
1	1	1

c	$f_{a=1,b=1}$
0	1
1	1

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

b	c	$f_{a=1}$
0	0	0
0	1	1
1	0	1
1	1	1

c	$f_{a=1,b=1}$
0	1
1	1

c	$f_{a=1,b=0}$
0	0
1	1

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

b	c	$f_{a=1}$
0	0	0
0	1	1
1	0	1
1	1	1

c	$f_{a=1,b=1}$
0	1
1	1

c	$f_{a=1,b=0}$
0	0
1	1

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

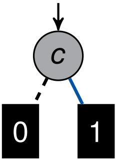
BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

b	c	$f_{a=1}$
0	0	0
0	1	1
1	0	1
1	1	1

c	$f_{a=1,b=1}$
0	1
1	1

c	$f_{a=1,b=0}$
0	0
1	1

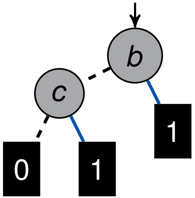


a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (1/3)

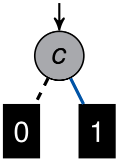
- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

b	c	$f_{a=1}$
0	0	0
0	1	1
1	0	1
1	1	1



c	$f_{a=1,b=1}$
0	1
1	1

c	$f_{a=1,b=0}$
0	0
1	1



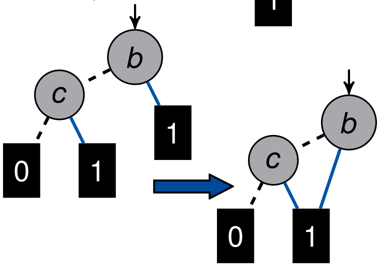
a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

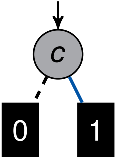
- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=1}(b,c) = b \cdot f_{a=1,b=1}(c) + b' \cdot f_{a=1,b=0}(c)$.

b	c	$f_{a=1}$
0	0	0
0	1	1
1	0	1
1	1	1

c	$f_{a=1,b=1}$
0	1
1	1

c	$f_{a=1,b=0}$
0	0
1	1





a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

b	c	$f_{a=0}$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

b	c	$f_{a=0}$
0	0	0
0	1	0
1	0	0
1	1	1

c	$f_{a=0,b=1}$
0	0
1	1

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

b	c	$f_{a=0}$
0	0	0
0	1	0
1	0	0
1	1	1

c	$f_{a=0,b=1}$
0	0
1	1

c	$f_{a=0,b=0}$
0	0
1	0

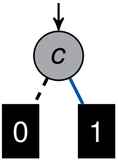
a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

b	c	$f_{a=0}$
0	0	0
0	1	0
1	0	0
1	1	1

c	$f_{a=0,b=1}$
0	0
1	1



c	$f_{a=0,b=0}$
0	0
1	0

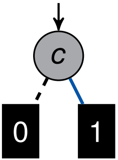
a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

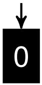
- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

b	c	$f_{a=0}$
0	0	0
0	1	0
1	0	0
1	1	1

c	$f_{a=0,b=1}$
0	0
1	1



c	$f_{a=0,b=0}$
0	0
1	0

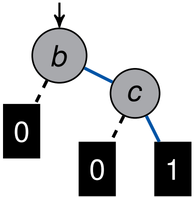


a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

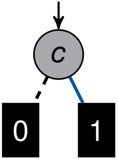
BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

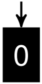
b	c	$f_{a=0}$
0	0	0
0	1	0
1	0	0
1	1	1



c	$f_{a=0,b=1}$
0	0
1	1



c	$f_{a=0,b=0}$
0	0
1	0

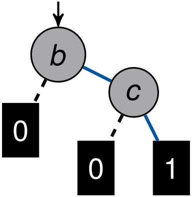


a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

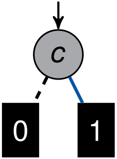
BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (2/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.
- $f_{a=0}(b,c) = b \cdot f_{a=0,b=1}(c) + b' \cdot f_{a=0,b=0}(c)$.

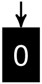
b	c	$f_{a=0}$
0	0	0
0	1	0
1	0	0
1	1	1

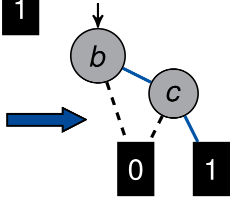


c	$f_{a=0,b=1}$
0	0
1	1



c	$f_{a=0,b=0}$
0	0
1	0





a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (3/3)

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

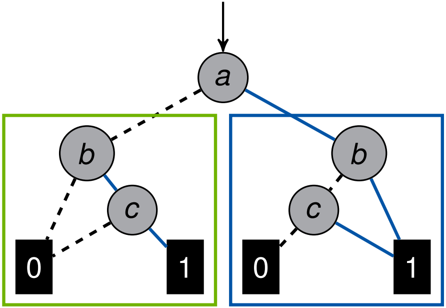
BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (3/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (3/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.



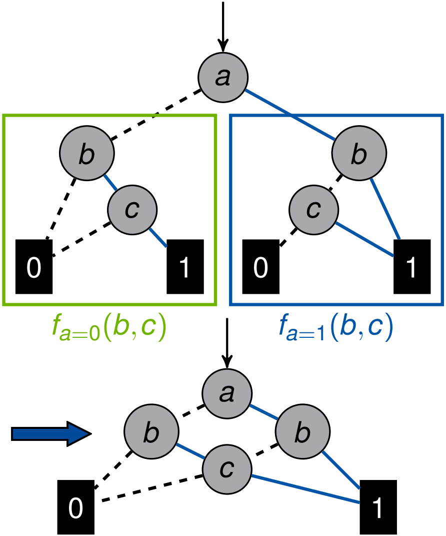
$f_{a=0}(b,c)$ $f_{a=1}(b,c)$

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

UNI FREIBURG

BDD des Carry-Bits $f := fa_1$ eines Volladierers aus Kofaktoren (3/3)

- $f(a,b,c) = a \cdot f_{a=1}(b,c) + a' \cdot f_{a=0}(b,c)$.

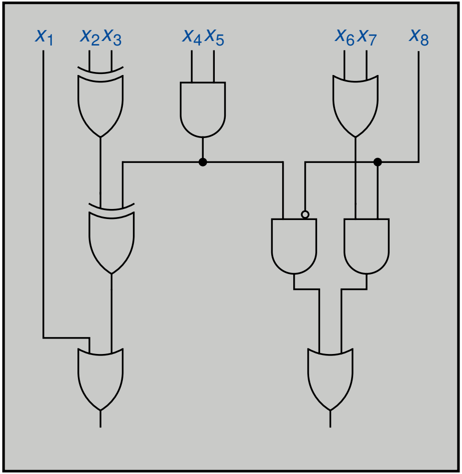


a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

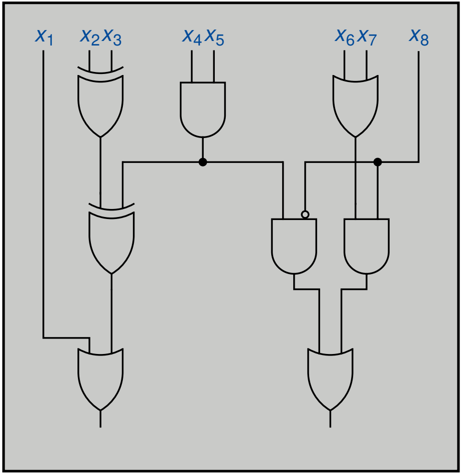
BDD aus einem booleschen Ausdruck

- Konstruiere BDD bdd_f von $f = x_1x_2x_3 + x_2'x_4 + x_3'x_4$.
  - Generiere bdd_{x_1}, bdd_{x_2}, bdd_{x_3}, bdd_{x_4}.
  - Berechne $bdd_{x_1x_2} := AND(bdd_{x_1}, bdd_{x_2})$.
  - Berechne $bdd_{x_1x_2x_3} := AND(bdd_{x_1x_2}, bdd_{x_3})$.
  - Berechne $bdd_{x_2'} := NOT(bdd_{x_2})$.
  - Berechne $bdd_{x_2'x_4} := AND(bdd_{x_2'}, bdd_{x_4})$.
  - Berechne $bdd_{x_3'} := NOT(bdd_{x_3})$.
  - Berechne $bdd_{x_3'x_4} := AND(bdd_{x_3'}, bdd_{x_4})$.
  - Berechne $bdd_{x_1x_2x_3+x_2'x_4} := OR(bdd_{x_1x_2x_3}, bdd_{x_2'x_4})$.
  - Berechne $bdd_f := OR(bdd_{x_1x_2x_3+x_2'x_4}, bdd_{x_3'x_4})$.
- Berechnung von AND, OR, NOT: Später!

BDD aus Schaltkreis: Symbolische Simulation



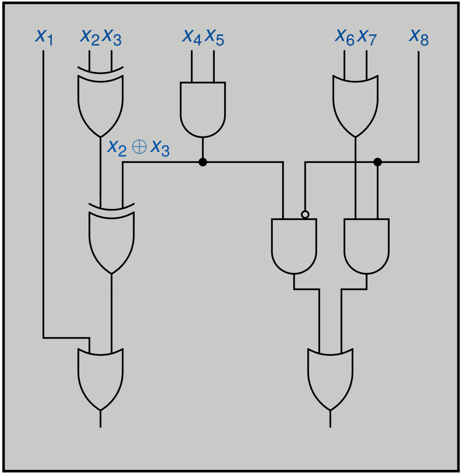
BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,

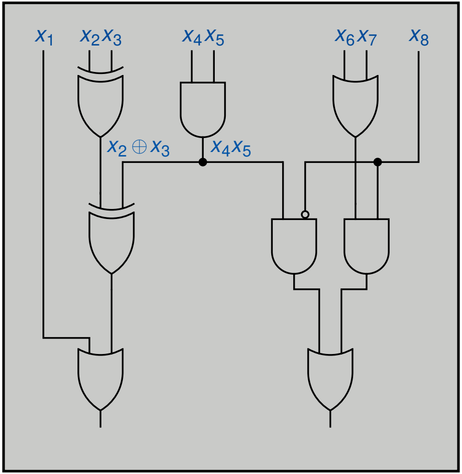
BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8},$
bdd_9
$\quad := XOR(bdd_{x_2}, bdd_{x_3}),$

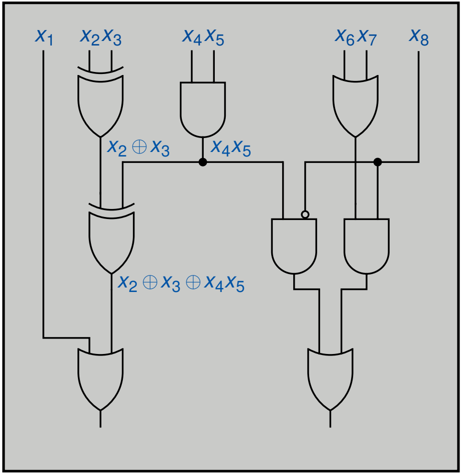
BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,
bdd_9
$\quad := XOR(bdd_{x_2}, bdd_{x_3})$,
bdd_{10}
$\quad := AND(bdd_{x_4}, bdd_{x_5})$,

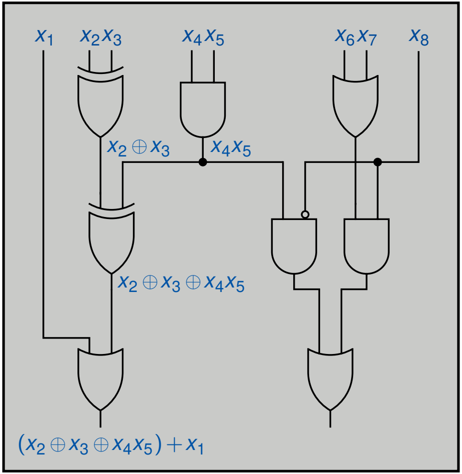
BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,
bdd_9
$:= XOR(bdd_{x_2}, bdd_{x_3})$,
bdd_{10}
$:= AND(bdd_{x_4}, bdd_{x_5})$,
bdd_{11}
$:= XOR(bdd_{x_9}, bdd_{x_{10}})$,

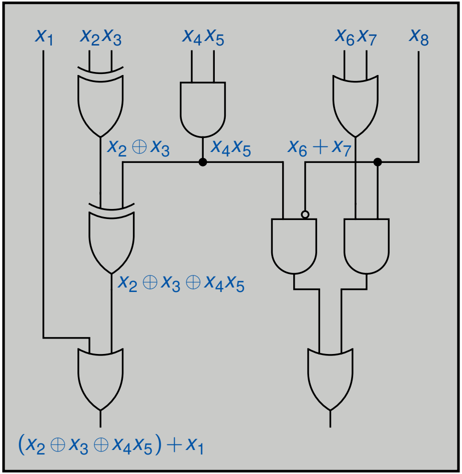
BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,
bdd_9
$:= XOR(bdd_{x_2}, bdd_{x_3})$,
bdd_{10}
$:= AND(bdd_{x_4}, bdd_{x_5})$,
bdd_{11}
$:= XOR(bdd_{x_9}, bdd_{x_{10}})$,
bdd_{out_1}
$:= OR(bdd_{x_1}, bdd_{x_{11}})$,

BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,

bdd_9
$:= XOR(bdd_{x_2}, bdd_{x_3})$,

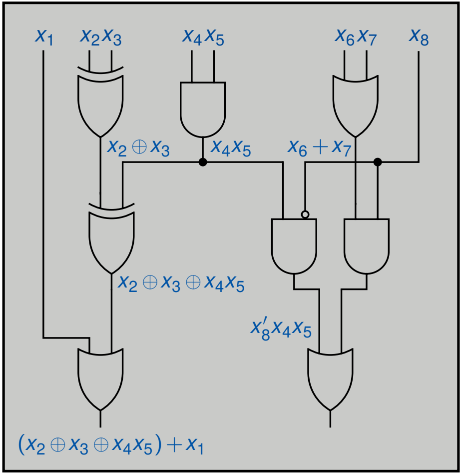
bdd_{10}
$:= AND(bdd_{x_4}, bdd_{x_5})$,

bdd_{11}
$:= XOR(bdd_{x_9}, bdd_{x_{10}})$,

bdd_{out_1}
$:= OR(bdd_{x_1}, bdd_{x_{11}})$,

bdd_{12}
$:= OR(bdd_{x_6}, bdd_{x_7})$,

BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,

bdd_9
$\quad := XOR(bdd_{x_2}, bdd_{x_3})$,

bdd_{10}
$\quad := AND(bdd_{x_4}, bdd_{x_5})$,

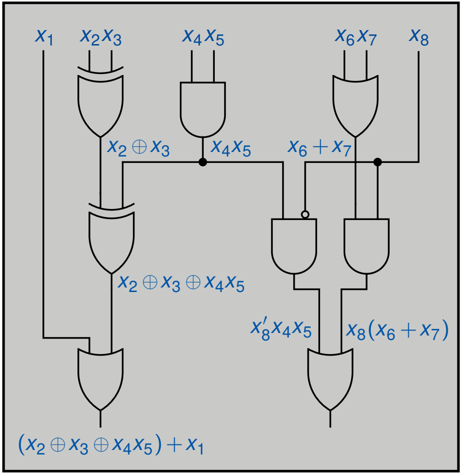
bdd_{11}
$\quad := XOR(bdd_{x_9}, bdd_{x_{10}})$,

bdd_{out_1}
$\quad := OR(bdd_{x_1}, bdd_{x_{11}})$,

bdd_{12}
$\quad := OR(bdd_{x_6}, bdd_{x_7})$,

bdd_{13}
$\quad := AND(bdd_{x_{10}}, NOT(bdd_{x_8}))$,

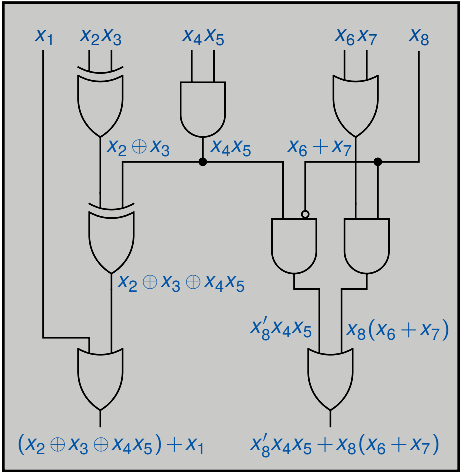
BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,
$bdd_9 := XOR(bdd_{x_2}, bdd_{x_3})$,
$bdd_{10} := AND(bdd_{x_4}, bdd_{x_5})$,
$bdd_{11} := XOR(bdd_{x_9}, bdd_{x_{10}})$,
$bdd_{out_1} := OR(bdd_{x_1}, bdd_{x_{11}})$,
$bdd_{12} := OR(bdd_{x_6}, bdd_{x_7})$,
$bdd_{13} := AND(bdd_{x_{10}}, NOT(bdd_{x_8}))$,
$bdd_{14} := AND(bdd_{x_{12}}, bdd_{x_8})$,

BDD aus Schaltkreis: Symbolische Simulation



BDD-Operationen:

$bdd_{x_1}, bdd_{x_2}, \ldots, bdd_{x_8}$,

$bdd_9 := XOR(bdd_{x_2}, bdd_{x_3})$,

$bdd_{10} := AND(bdd_{x_4}, bdd_{x_5})$,

$bdd_{11} := XOR(bdd_{x_9}, bdd_{x_{10}})$,

$bdd_{out_1} := OR(bdd_{x_1}, bdd_{x_{11}})$,

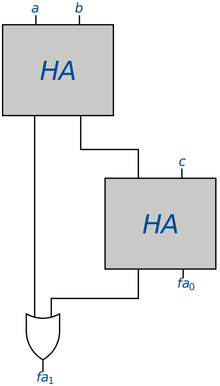
$bdd_{12} := OR(bdd_{x_6}, bdd_{x_7})$,

$bdd_{13} := AND(bdd_{x_{10}}, NOT(bdd_{x_8}))$,

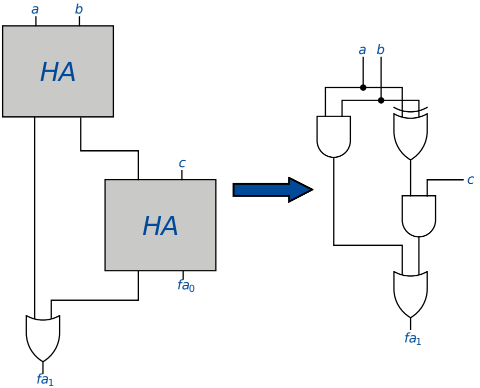
$bdd_{14} := AND(bdd_{x_{12}}, bdd_{x_8})$,

$bdd_{out_2} := OR(bdd_{x_{13}}, bdd_{x_{14}})$.

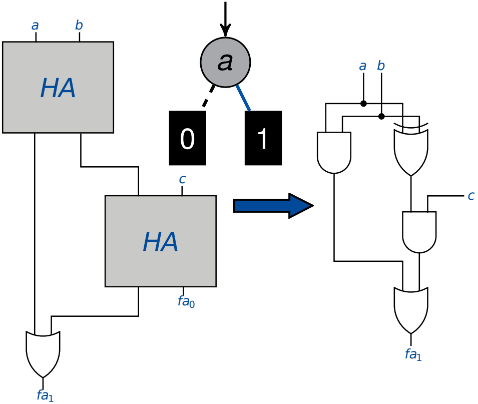
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



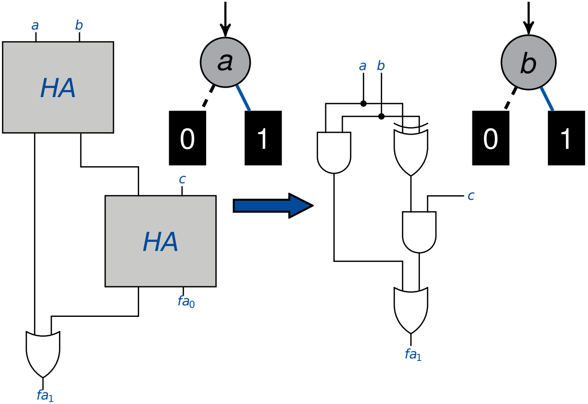
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



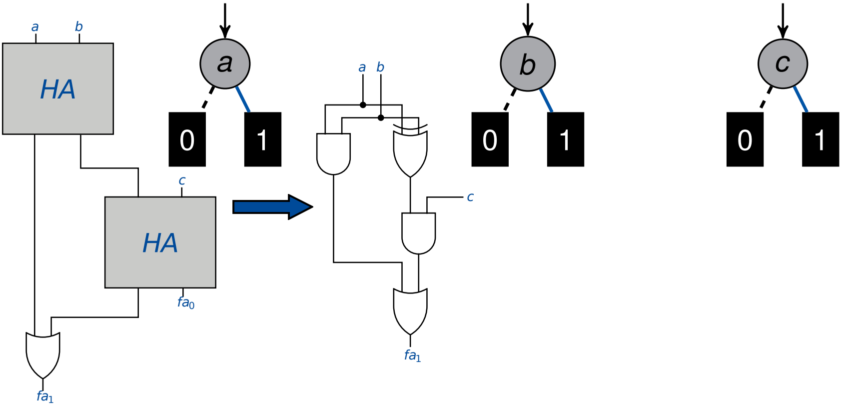
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



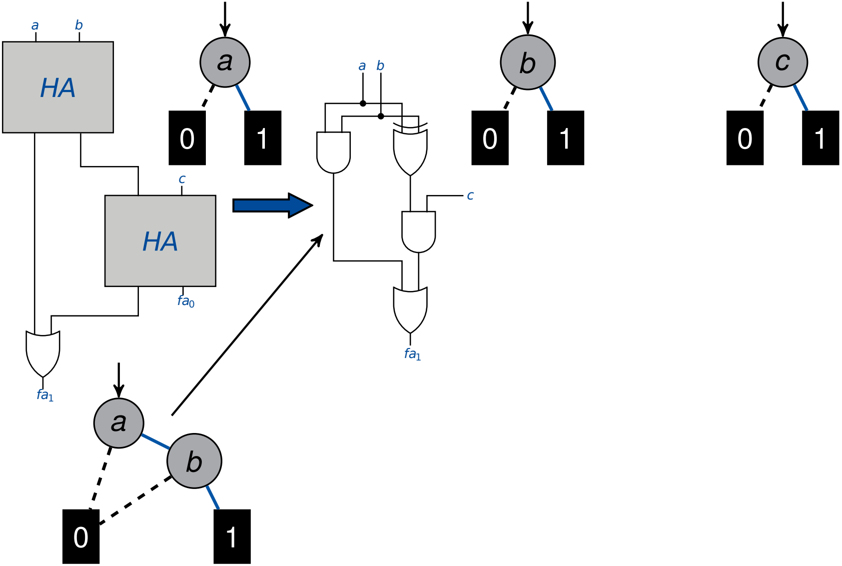
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



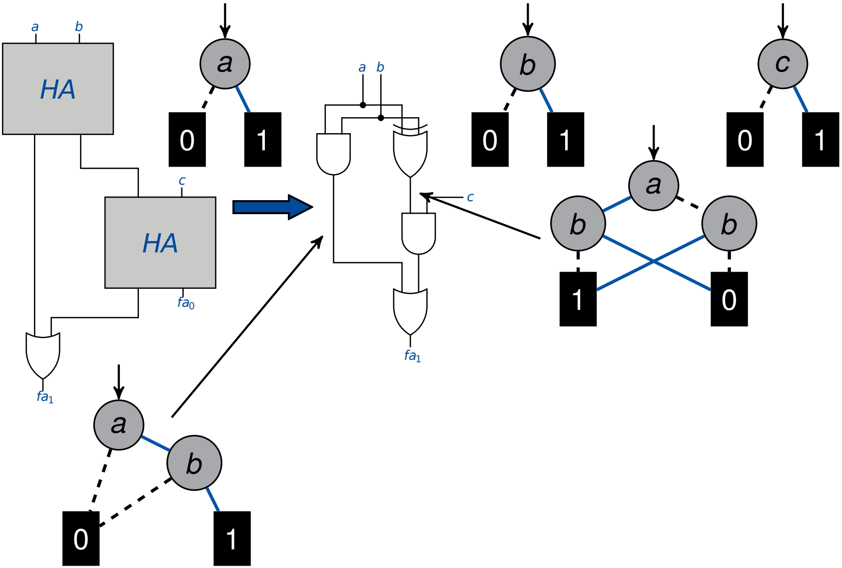
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



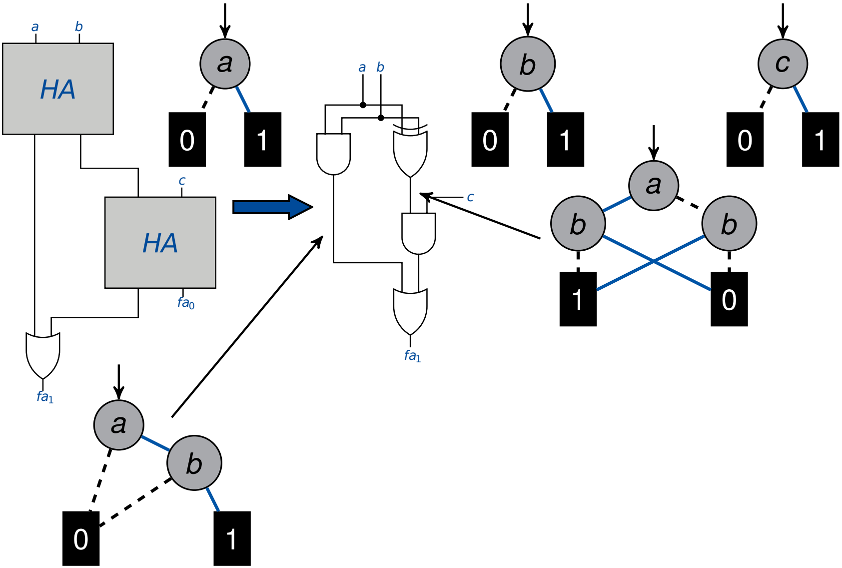
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



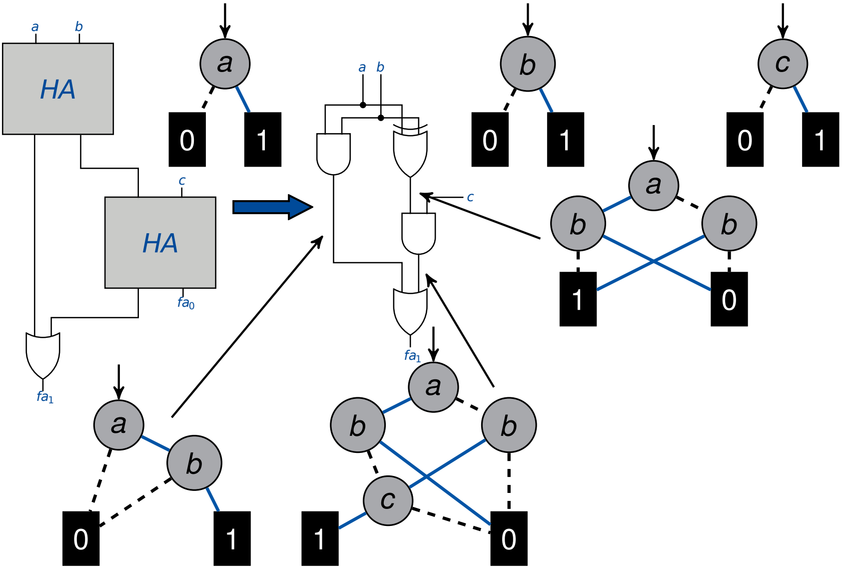
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



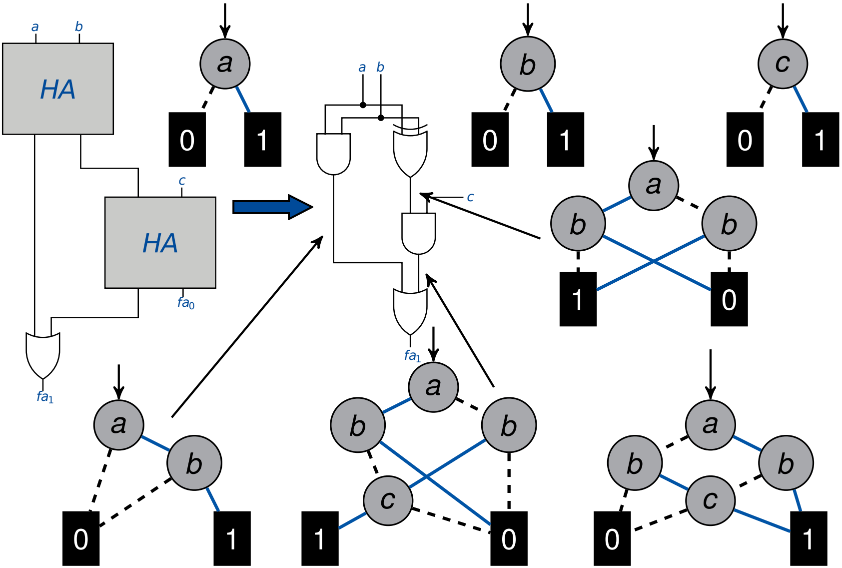
Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis



Beispiel: BDD des Carry-Bits $f := fa_1$ eines FAs aus Schaltkreis

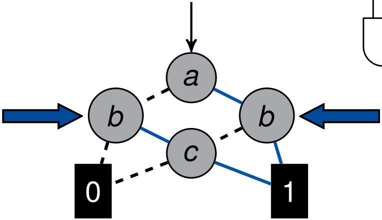


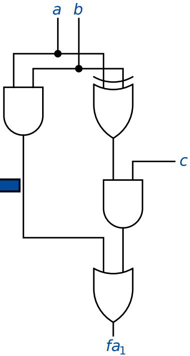
UNI FREIBURG

Übrigens …

- Wir haben soeben einen Äquivalenzcheck durchgeführt!

a	b	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





Realisierung von booleschen Operationen zwischen BDDs

- Gegeben BDDs a und b, berechne BDD $c = AND(a,b)$.
- Erste Option: Berechne die Funktionstabelle von $(a \cdot b)$, erzeuge daraus das BDD mit der Kofaktor-Methode $\rightarrow$ sehr ineffizient!
- Besserer Ansatz: If-Then-Else-Funktion (*ITE*).
  - $ITE(F,G,H) := FG + F'H$.
  - Alle binären Operationen sind auf *ITE* zurückführbar.
    - $AND(F,G) = ITE(F,G,0)$
    - $OR(F,G) = ITE(F,1,G)$
    - $NOT(F) = ITE(F,0,1)$
    - $XOR(F,G) = ITE(F,G',G)$

Realisierung von booleschen Operationen zwischen BDDs

- Gegeben BDDs a und b, berechne BDD $c = AND(a,b)$
- Erste Option: Berechne die Funktionstabelle von $(a \cdot b)$, erzeuge daraus das BDD mit der Kofaktor-Methode $\rightarrow$ sehr ineffizient!
- Besserer Ansatz: If-Then-Else-Funktion (*ITE*).
  - $ITE(F,G,H) := FG + F'H$.
  - Alle binären Operationen sind auf *ITE* zurückführbar.
    - $AND(F,G) = ITE(F,G,0)$
    - $OR(F,G) = ITE(F,1,G)$
    - $NOT(F) = ITE(F,0,1)$
    - $XOR(F,G) = ITE(F,G',G)$

Berechnung von *ITE* auf BDDs-Prinzip

- Gegeben sind drei BDDs F, G und H mit der gleichen Variablenordnung. Gesucht ist BDD für $ITE(F,G,H)$.
- Es gilt für eine Variable x:
- $ITE(F,G,H)$
  $= FG + F'H$
  $= x(FG+F'H)_{x=1} + x'(FG+F'H)_{x=0}$
  $= x(F_{x=1}G_{x=1} + (F_{x=1})'H_{x=1}) + x'(F_{x=0}G_{x=0} + (F_{x=0})'H_{x=0})$
  $= x \cdot ITE(F_{x=1}, G_{x=1}, H_{x=1}) + x' \cdot ITE(F_{x=0}, G_{x=0}, H_{x=0})$.
- *ITE*-Berechnung im vollständigen BDD:
  - Sei x die oberste Variable in BDDs F, G und H.
  - BDDs für $F_{x=1}, G_{x=1}, H_{x=1}, F_{x=0}, G_{x=0}, H_{x=0}$ sind Kinder von x.
  - Berechne BDDs $ITE(F_{x=1}, G_{x=1}, H_{x=1})$ und $ITE(F_{x=0}, G_{x=0}, H_{x=0})$ rekursiv. x mit diesen BDDs als Kinder ist das gesuchte BDD.

ITE auf reduzierten BDDs

- $\mathit{ITE}(F,G,H)$
  $= x \cdot \mathit{ITE}(F_{x=1}, G_{x=1}, H_{x=1}) + x' \cdot \mathit{ITE}(F_{x=0}, G_{x=0}, H_{x=0})$.

- Rekursive Vorgehensweise:

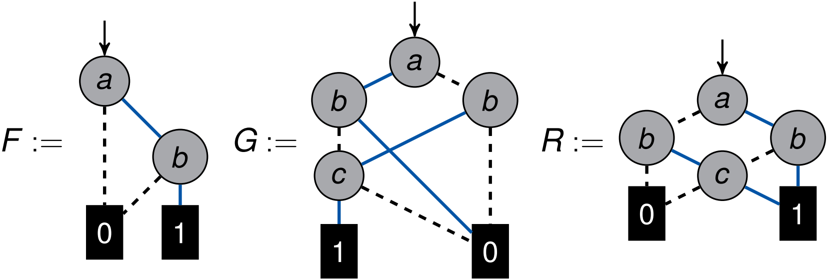
  - Berechne BDDs $\mathit{ITE}(F_{x=1}, G_{x=1}, H_{x=1})$ und $\mathit{ITE}(F_{x=0}, G_{x=0}, H_{x=0})$.

  - Falls $\mathit{ITE}(F_{x=1}, G_{x=1}, H_{x=1}) = \mathit{ITE}(F_{x=0}, G_{x=0}, H_{x=0})$, so ist $\mathit{ITE}(F,G,H) = \mathit{ITE}(F_{x=1}, G_{x=1}, H_{x=1}) = \mathit{ITE}(F_{x=0}, G_{x=0}, H_{x=0})$.

  - Sonst generiere einen neuen Knoten x, dessen high-Kante auf $\mathit{ITE}(F_{x=1}, G_{x=1}, H_{x=1})$ und low-Kante auf $\mathit{ITE}(F_{x=0}, G_{x=0}, H_{x=0})$ zeigt.

  - **Terminalfälle**:
    $\mathit{ITE}(1,F,G) = \mathit{ITE}(0,G,F) = \mathit{ITE}(F,1,0) = \mathit{ITE}(G,F,F) = F$.

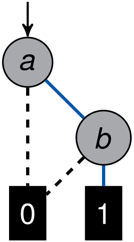
ITE-Berechnung: Beispiel (1/2)



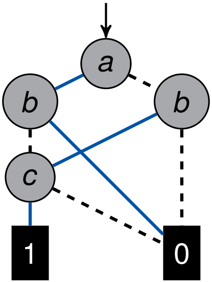
- Wir haben vorhin gerechnet: $R = OR(F, G)$
- Dies entspricht: $R = ITE(F, 1, G)$.
- Wir wollen nun die rekursive *ITE*-Berechnung nachvollziehen.
- Es liegt kein Terminalfall vor, obere Variable: a.

ITE-Berechnung: Beispiel (2/2)

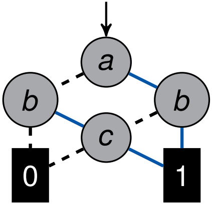
$F :=$



$G :=$

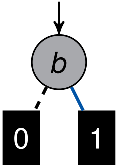


$R :=$



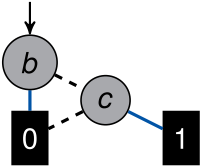
- Berechne zunächst rekursiv $ITE(F_{a=1}, 1_{a=1}, G_{a=1})$.

$F_{a=1} =$



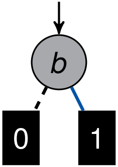
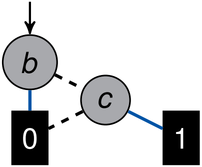
$, 1_{a=1} = 1,$

$G_{a=1} =$

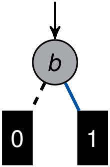


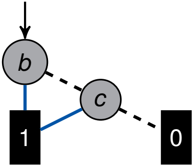
- Rechne rekursiv mit Variable b weiter:
  $ITE(F_{a=1}, 1_{a=1}, G_{a=1}) = b \cdot ITE(F_{a=1,b=1}, 1_{a=1,b=1}, G_{a=1,b=1}) +$
  $b' \cdot ITE(F_{a=1,b=0}, 1_{a=1,b=0}, G_{a=1,b=0})$.

Beispiel: $ITE(F_{a=1}, 1_{a=1}, G_{a=1})$

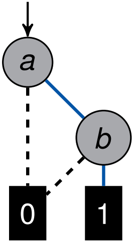
$F_{a=1} =$  $, 1_{a=1} = 1,$ $G_{a=1} =$ 

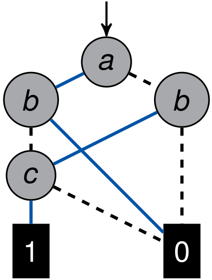
- $ITE(F_{a=1}, 1_{a=1}, G_{a=1}) = b \cdot ITE(F_{a=1,b=1}, 1_{a=1,b=1}, G_{a=1,b=1})$ $+ b' \cdot ITE(F_{a=1,b=0}, 1_{a=1,b=0}, G_{a=1,b=0})$.
  - $F_{a=1,b=1} = 1_{a=1,b=1} = 1, G_{a=1,b=1} = 0,$ $ITE(1,1,0) = 1$ **(Terminalfall)**.
  - $F_{a=1,b=0} = 0, 1_{a=1,b=0} = 1, G_{a=1,b=0} = c =$ $ITE(0,1,c) = c$ **(Terminalfall)**.

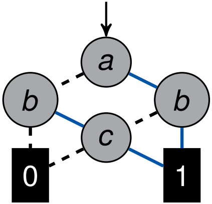


- Insgesamt: $ITE(F_{a=1}, 1_{a=1}, G_{a=1}) =$ 

Beispiel: $ITE(F_{a=0}, 1_{a=0}, G_{a=0})$

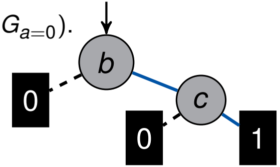
$F :=$ 

$G :=$ 

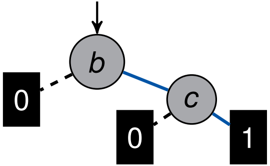
$R :=$ 

- Berechne zunächst rekursiv $ITE(F_{a=0}, 1_{a=0}, G_{a=0})$.

$F_{a=0} = 0, \quad 1_{a=0} = 1, \quad G_{a=0} =$



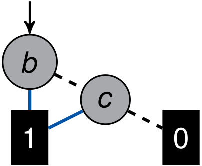
Ein **Terminalfall** liegt vor:
$ITE(0, 1, G_{a=0}) = G_{a=0} =$



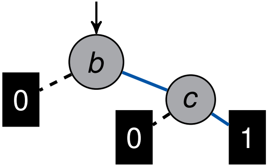
Beispiel - Ende

- $ITE(F,1,G) = a \cdot ITE(F_{a=1}, 1_{a=1}, G_{a=1}) + a' \cdot ITE(F_{a=0}, 1_{a=0}, G_{a=0})$.

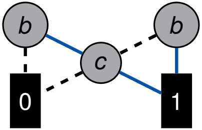
$ITE(F_{a=1}, 1_{a=1}, G_{a=1}) =$



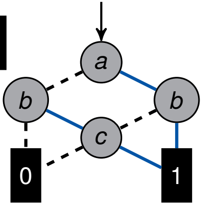
$ITE(F_{a=0}, 1_{a=0}, G_{a=0}) =$



- Durch die Verwendung der sogenannten Computed Tables (s.u.) sind die berechneten BDDs bereits reduziert:



- Somit muss *ITE* nur noch einen Knoten *a* einfügen und seine ausgehenden Kanten „richtig lenken".



Zur Komplexität von *ITE*

- Bei der rekursiven Konstruktion von *ITE* überprüft man, ob die BDDs $ITE(F_{x=1}, G_{x=1}, H_{x=1})$ und $ITE(F_{x=0}, G_{x=0}, H_{x=0})$ vielleicht bereits generiert wurden und in der sogenannten Computed Table (CT) vorliegen.

- Nur wenn dies nicht der Fall ist, werden die BDDs erzeugt und in die CT eingefügt.

- Implementiert man dies effizient, wird die Komplexität von *ITE* zu $O(|F| \cdot |G| \cdot |H|)$.
  - Ohne Beweis.

- Somit lässt sich etwa die Komplexität von $AND(F,G) = ITE(F,G,0)$ durch $O(|F| \cdot |G|)$ abschätzen (0 ist das BDD für die konstante 0-Funktion, das aus 1 Knoten besteht.)

ITE-Algorithmus

```
ITE(F,G,H){
    if (F == 1) return G;      // Überprüfe erst 4 Terminalfälle.
    if (F == 0) return H;
    if ((G == 1) and (H == 0)) return F;
    if (G == H) return G;

    x = Top_Variable(F,G,H);      // Gleiche Variablenordnung.
    high = ITE(F_{x=1}, G_{x=1}, H_{x=1});
    low = ITE(F_{x=0}, G_{x=0}, H_{x=0});
    if (high == low) return low;
    R = Find_Or_Add_in_Computed_Table(x, high, low);
    return R;
}
```

Größe von BDDs

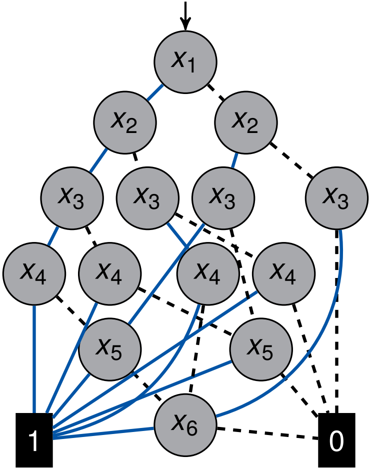
- Oft exponentiell.
- Hängt stark von der Variablenordnung ab.
  - Siehe Beispiel auf der nächsten Folie.
- Für viele in der Praxis vorkommende Funktionen sind BDDs zumindest für einige Variablenordnungen kompakt.
- BDD-Minimierung: Finden einer „guten" Variablenordnung (NP-vollständiges Problem).
  - Exakte und heuristische Verfahren.
  - Für einige Funktionen, u.a. Multiplizierer, ist BDD-Größe für jede Variablenordnung exponentiell.

Einfluss der Variablenordnung

- **Beispiel**: $f^n(x_1,\ldots,x_{2n}) = x_1 x_{n+1} + x_2 x_{n+2} + \cdots + x_n x_{2n}$.

Einfluss der Variablenordnung

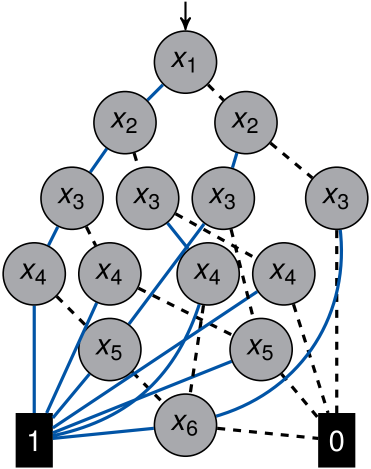
- **Beispiel**: $f^n(x_1,\ldots,x_{2n}) = x_1x_{n+1} + x_2x_{n+2} + \cdots + x_nx_{2n}$.



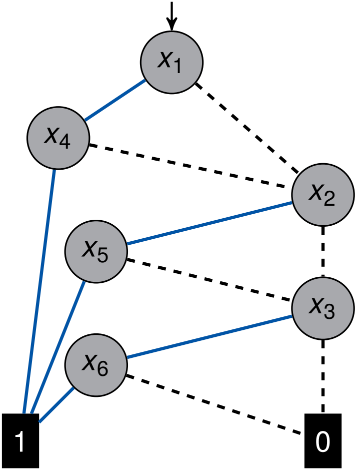
exponentielle Größe

Einfluss der Variablenordnung

- **Beispiel**: $f^n(x_1, \dots, x_{2n}) = x_1 x_{n+1} + x_2 x_{n+2} + \cdots + x_n x_{2n}$.

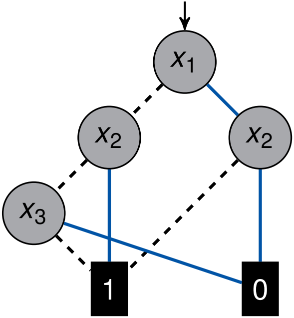


exponentielle Größe

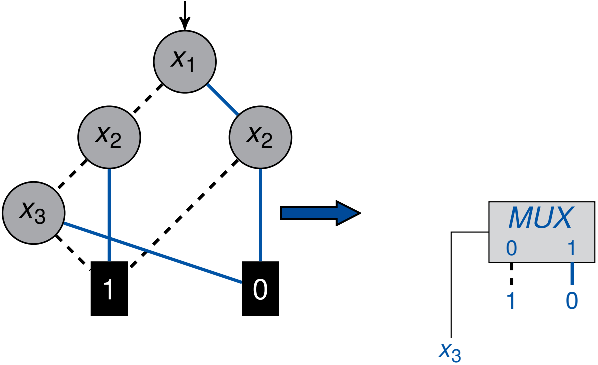


lineare Größe

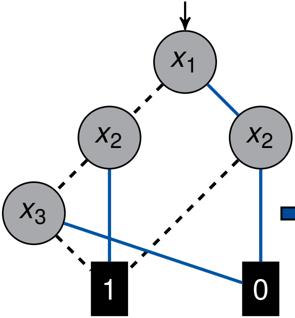
Schaltkreis direkt aus BDD

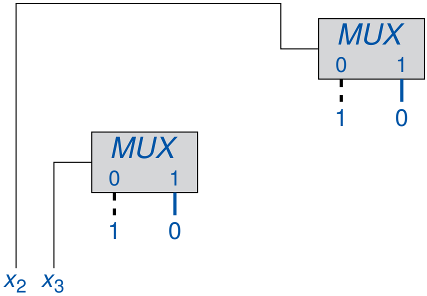


Schaltkreis direkt aus BDD

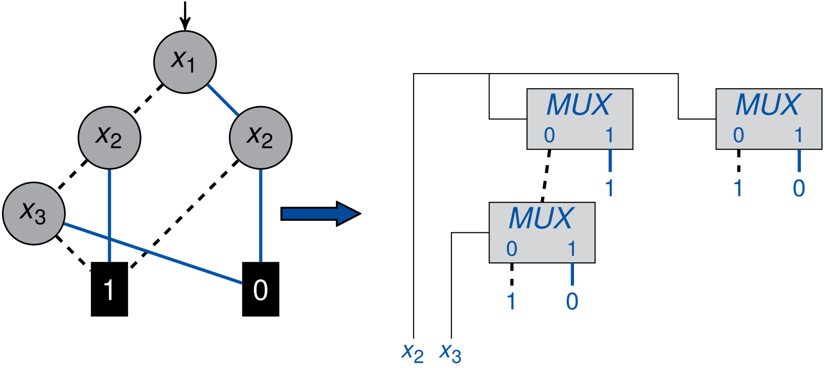


Schaltkreis direkt aus BDD

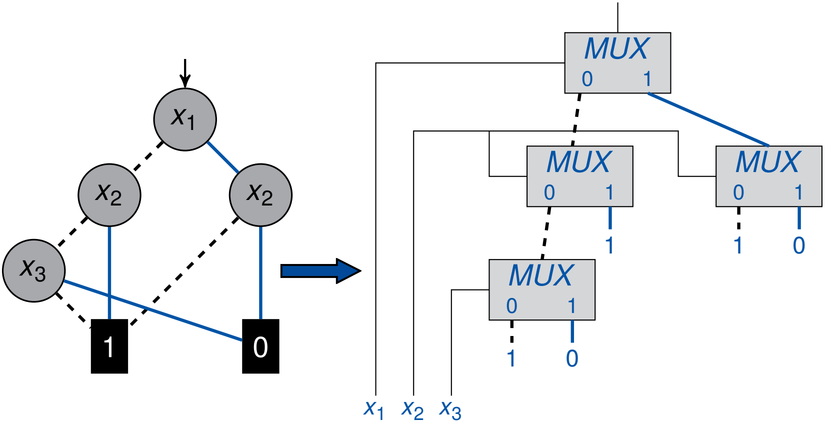




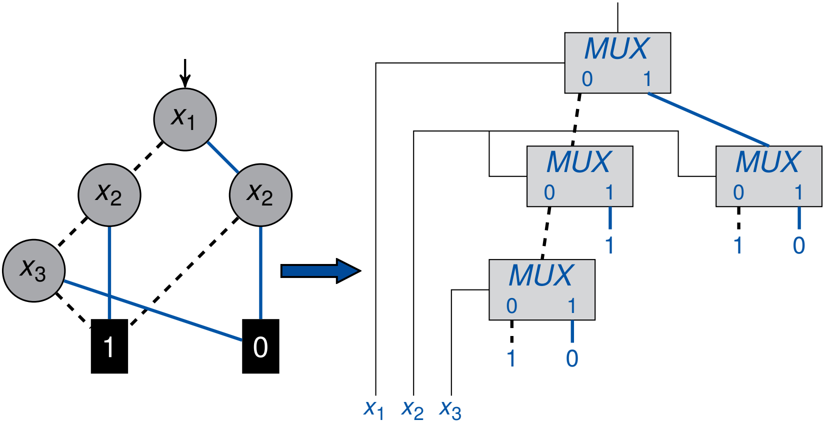
Schaltkreis direkt aus BDD



Schaltkreis direkt aus BDD



Schaltkreis direkt aus BDD



… beschreibt die boolesche Funktion $x_1'x_2'x_3' + x_1'x_2 + x_1x_2'$.