Technische Informatik 1

Prof. Dr. Rolf Drechsler

Christina Plump

Überblick

Teil 1: Der Rechneraufbau (Kapitel 2-5)

- Rechner im Überblick
- Pipelining
- Speicher
- Parallelverarbeitung

Teil 2: Der Funktionalitätsaufbau (Kapitel 6-12)

- Kodierung
- Grundbegriffe, Boolesche Funktionen
- **Darstellungsmöglichkeiten**
  - Zweistufige Logiksynthese
  - **Binäre Entscheidungsdiagramme**
- Schaltkreise, Synthese, spezielle Schaltkreise

Kapitel 9: Decision Diagrams

Binary Decision Diagrams (BDDs)

Weitere DD-Typen

Lernziele

- Decision Diagrams als alternative Repräsentationsmöglichkeit für Boolesche Funktionen kennenlernen
- Syntax und Semantik von Binary Decision Diagrams kennenlernen und verstehen
- Zerlegungsverfahren nach Shannon kennen und anwenden können
- Kriterien für ein reduziertes Binary Decision Diagram kennen und anwenden können
- Eindeutigkeit von reduzierten geordneten Binary Decision Diagrams verstehen
- Einfluss der Variablenordnung auf die Größe eines Binary Decision Diagrams kennen und verstehen
- ITE-Operator kennen, verstehen und anwenden können

Decision Diagrams – Motivation & Ziele

- Effiziente Darstellung Boolescher Funktionen
  - erstes Beispiel mehrstufige Schaltkreise
  - wichtig als Datenstruktur
    - für die Logiksynthese
    - für die Verifikation
- Gelungener Kompromiss zwischen kompakter Darstellung und effizienter Manipulation
- Kanonische Darstellung
  - Äquivalenzprüfung reduziert zu Gleichheit
  - Tautologietest einfach durchführbar

Decision Diagrams – Die Familie

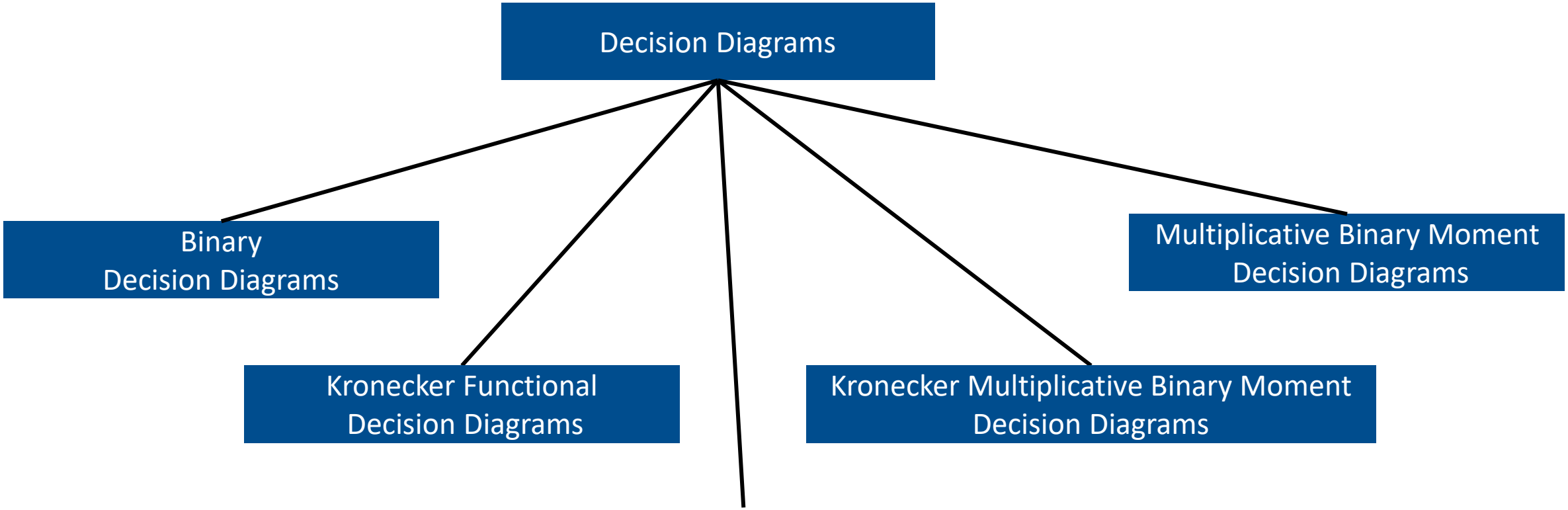
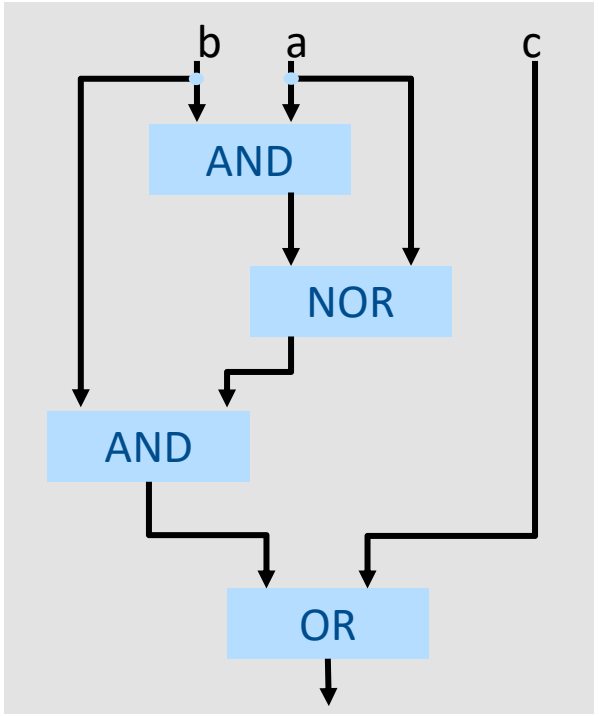
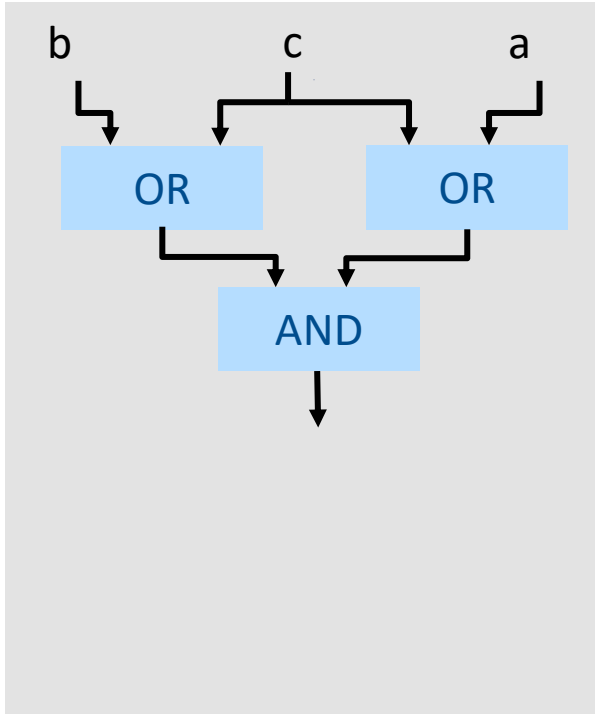


Illustration der Idee

$C_1 :=$



?
$\equiv$



$=: C_2$

Normalform der durch C_1 realisierten Booleschen Funktion

?
$=$

Normalform der durch C_2 realisierten Booleschen Funktion

Normalformen Boolescher Funktionen

- Kanonische Disjunktive Normalform $\overline{a} \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot c + a \cdot b \cdot \overline{c} + a \cdot b \cdot c$
- Kanonische Konjunktive Normalform $(a + b + c) \cdot \left(a + \overline{b} + c\right) \cdot (\overline{a} + b + c)$
- Ringsummennormalform $c \oplus (a \cdot b) \oplus (a \cdot b \cdot c)$

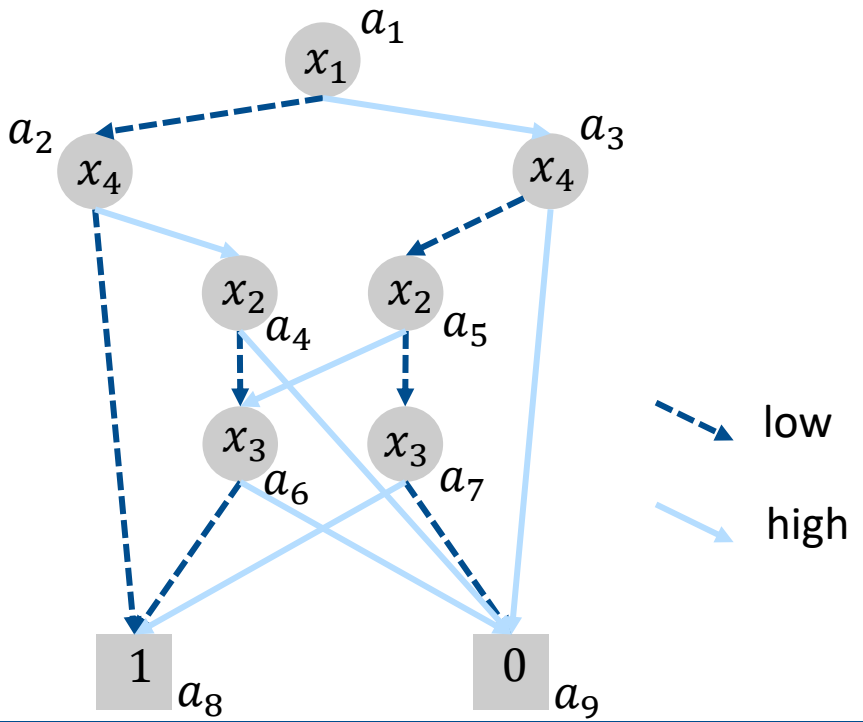
sind aufgrund ihrer sehr großen Kosten in der Praxis nur für Boolesche Funktionen mit wenigen Eingängen einsetzbar.

Decision Diagrams (DDs) als Alternative

Definition (Decision Diagram):
Ein Decision Diagram ist ein zykelfreier gerichteter Graph mit einer Wurzel und einer Label-Funktion. Innere Knoten tragen eine Variable als Label und haben zwei Nachfolger: „low-Kind“ und „high-Kind“. Blätter sind mit einer Konstanten markiert.

Beispiel:

- $G = (V = V_n \cup V_t, E, l: V \to X_n)$
- $V_n = \{a_1, \dots, a_7\}, V_t = \{a_8, a_9\}$
- $E = \{(a_1, a_2), (a_1, a_3), (a_2, a_4), (a_3, a_5), (a_2, a_8),$
  $(a_4, a_6), (a_4, a_9), (a_5, a_6), (a_5, a_7), (a_3, a_9),$
  $(a_6, a_8), (a_6, a_9), (a_7, a_8), (a_7, a_9)\}$
- $l(a_1) = x_1,$
- $l(a_2) = l(a_3) = x_4,$
- $l(a_4) = l(a_5) = x_2,$
- $l(a_6) = l(a_7) = x_3,$
- $l(a_8) = 1, l(a_9) = 0$

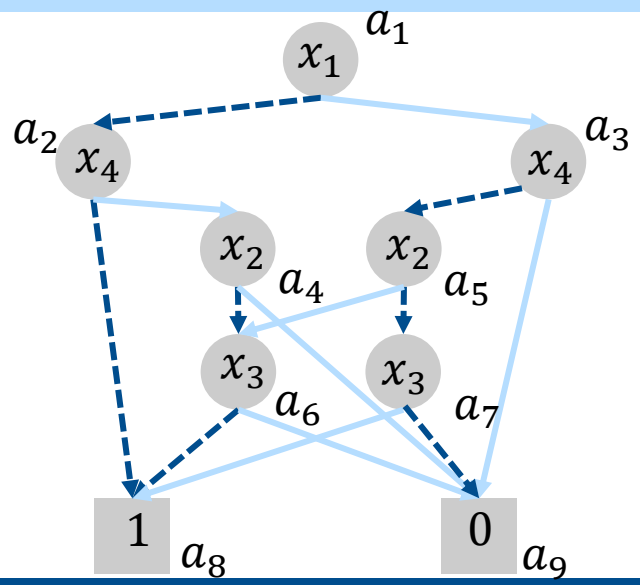


Decision Diagrams (DDs) als Alternative

Definition (Binary Decision Diagram über X_n - Syntax):
Ein Binary Decision Diagram ist ein Decision Diagram mit Konstanten 0 und 1.

Definition (Binary Decision Diagram über X_n - Semantik):
Die Funktion f_v an einem inneren Knoten $v \in V_n$ mit $l(v) = x_i \in X_n$ entspricht: $f_v = x_i \cdot f_{high(v)} + \overline{x_i} \cdot f_{low(v)}$. Die Funktion f_v an einem Blattknoten $v \in V_t$ mit $l(v) = c \in \{0,1\}$ entspricht dem Label selbst, d.h. $f_v = c$.

- $f_{a_8} = 1, f_{a_9} = 0$
- $f_{a_7} = x_3 \cdot f_{a_8} + \overline{x_3} \cdot f_{a_9} = x_3 \cdot 1 + \overline{x_3} \cdot 0 = x_3$
- $f_{a_6} = x_3 \cdot f_{a_9} + \overline{x_3} \cdot f_{a_8} = x_3 \cdot 0 + \overline{x_3} \cdot 1 = \overline{x_3}$
- $f_{a_5} = x_2 \cdot f_{a_6} + \overline{x_2} \cdot f_{a_7} = x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 = x_2 \oplus x_3$
- $f_{a_4} = x_2 \cdot f_{a_9} + \overline{x_2} \cdot f_{a_6} = x_2 \cdot 0 + \overline{x_2} \cdot \overline{x_3} = \overline{x_2} \cdot \overline{x_3}$
- $f_{a_3} = x_4 \cdot f_{a_9} + \overline{x_4} \cdot f_{a_5} = x_4 \cdot 0 + \overline{x_4}(x_2 \oplus x_3) = \overline{x_4}(x_2 \oplus x_3)$
- $f_{a_2} = x_4 \cdot f_{a_4} + \overline{x_4} \cdot f_{a_8} = x_4 \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_4} \cdot 1 = x_4 \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_4}$
- $f_{a_1} = x_1 \cdot f_{a_3} + \overline{x_1} \cdot f_{a_2} = x_1 \cdot \overline{x_4}(x_2 \oplus x_3) + \overline{x_1} \cdot (x_4 \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_4})$



Kofaktoren und Shannon-Zerlegung

Definition (Kofaktor):
Sei $f \in \mathcal{B}_n$ eine Boolesche Funktion in n Variablen. Dann heißt die Funktion $f_{x_i=1}: \mathbb{B}^n \to \mathbb{B}$ mit $f_{x_i=1}(\alpha_1, \dots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \dots, \alpha_n) = f(\alpha_1, \dots, \alpha_{i-1}, 1, \alpha_{i+1}, \dots, \alpha_n)$ für alle $\alpha \in \{0,1\}^n$ Kofaktor von f nach $x_i = 1$.
Entsprechend heißt $f_{x_i=0}$ der Kofaktor von f nach $x_i = 0$.

Lemma:
Sei $f \in \mathcal{B}_n$ eine Boolesche Funktion in n Variablen. Dann gilt $f = \left(x_i \cdot f_{x_i=1}\right) + \left(\overline{x_i} \cdot f_{x_i=0}\right)$.

Beweis: erfolgt durch Nachrechnen

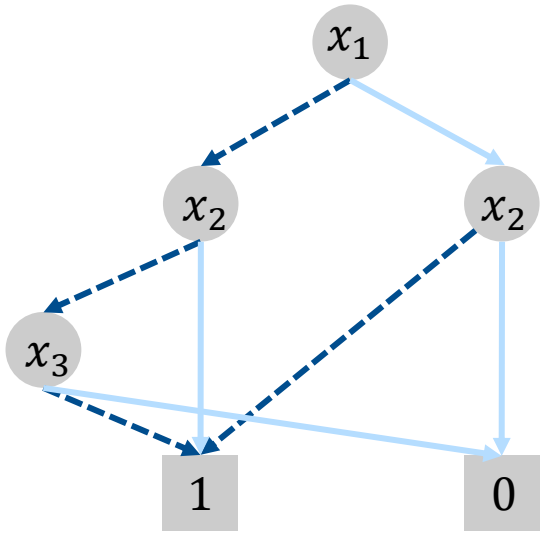
Definition (Shannon-Zerlegung):
Sei $f \in \mathcal{B}_n$ eine Boolesche Funktion in n Variablen. Für $i \in \{1, \dots, n\}$ heißt $f = \left(x_i \cdot f_{x_i=1}\right) + \left(\overline{x_i} \cdot f_{x_i=0}\right)$ Shannon-Zerlegung von f.

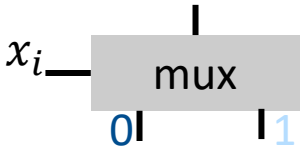
Davio-Zerlegung

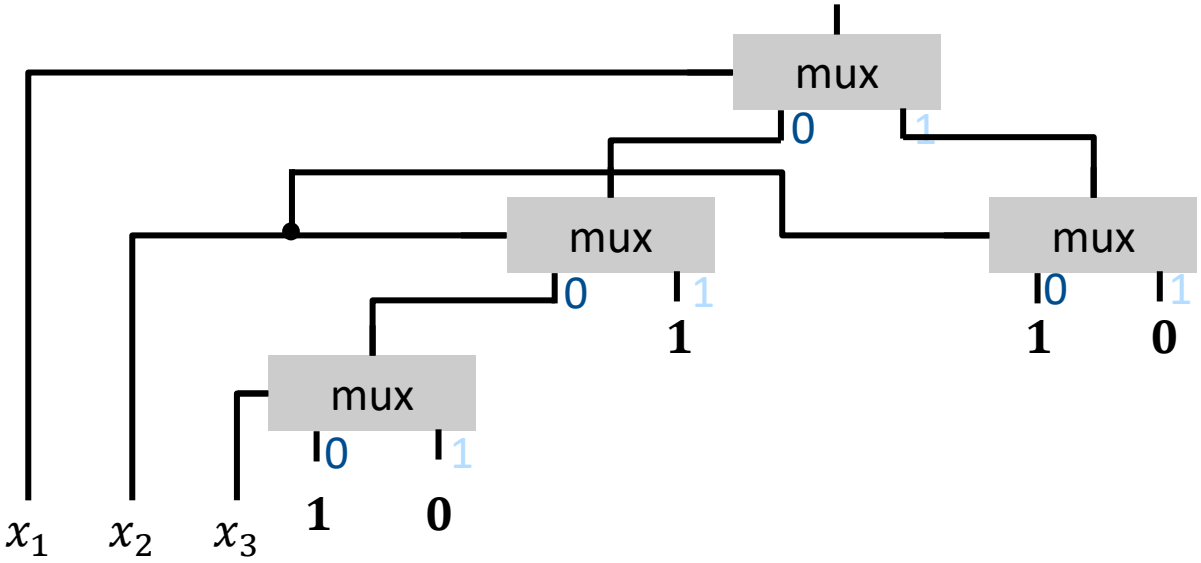
Definition (positive Davio-Zerlegung):
Sei $f \in \mathcal{B}_n$ eine Boolesche Funktion in n Variablen. Für $i \in \{1, \dots, n\}$ heißt $f = f_{x_i=0} \oplus x_i \cdot (f_{x_i=0} \oplus f_{x_i=1})$ positive Davio-Zerlegung von f.

Definition (negative Davio-Zerlegung):
Sei $f \in \mathcal{B}_n$ eine Boolesche Funktion in n Variablen. Für $i \in \{1, \dots, n\}$ heißt $f = f_{x_i=1} \oplus \overline{x_i} \cdot (f_{x_i=0} \oplus f_{x_i=1})$ positive Davio-Zerlegung von f.

Interpretation von BDDs als mehrstufiger Schaltkreis

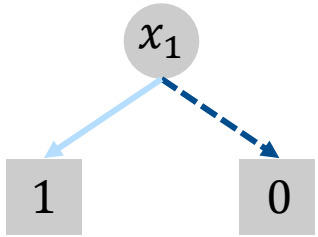




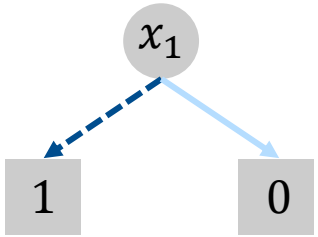


Beschreibt die Boolesche Funktion $\overline{x_1}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}\,x_2 + x_1\overline{x_2}$

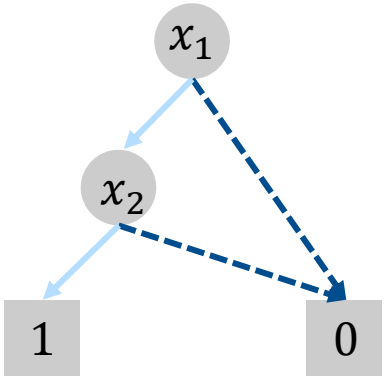
BDDs von typischen Funktionen



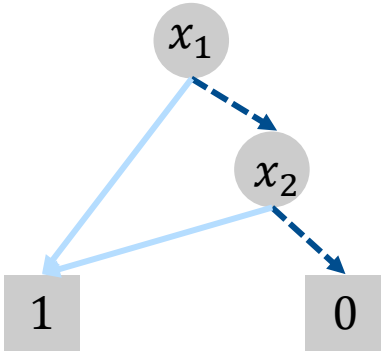
$f(x_1) = x_1$



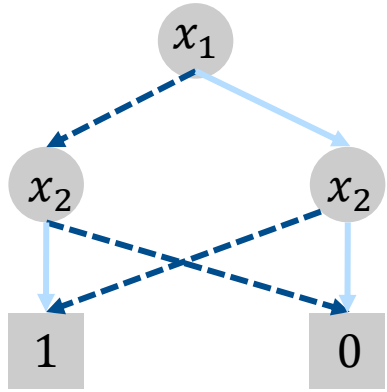
$f(x_1) = \overline{x_1}$



$f(x_1, x_2) = x_1 \cdot x_2$



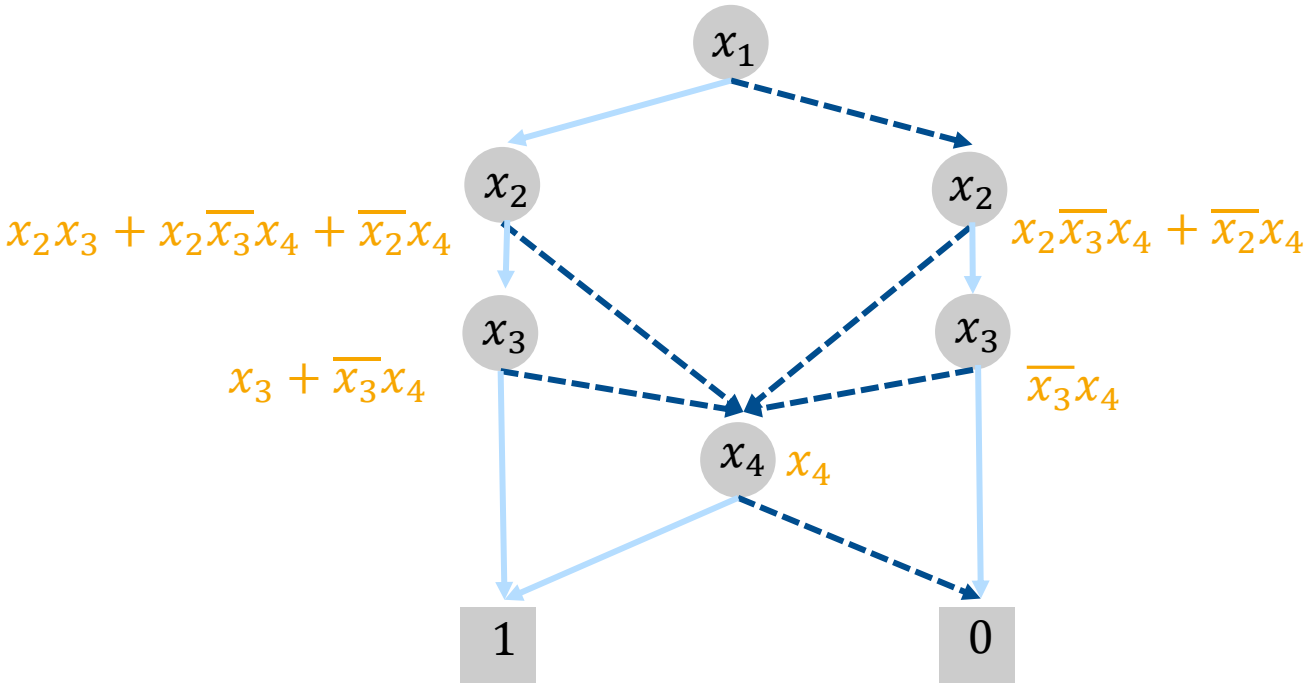
$f(x_1, x_2) = x_1 + x_2$



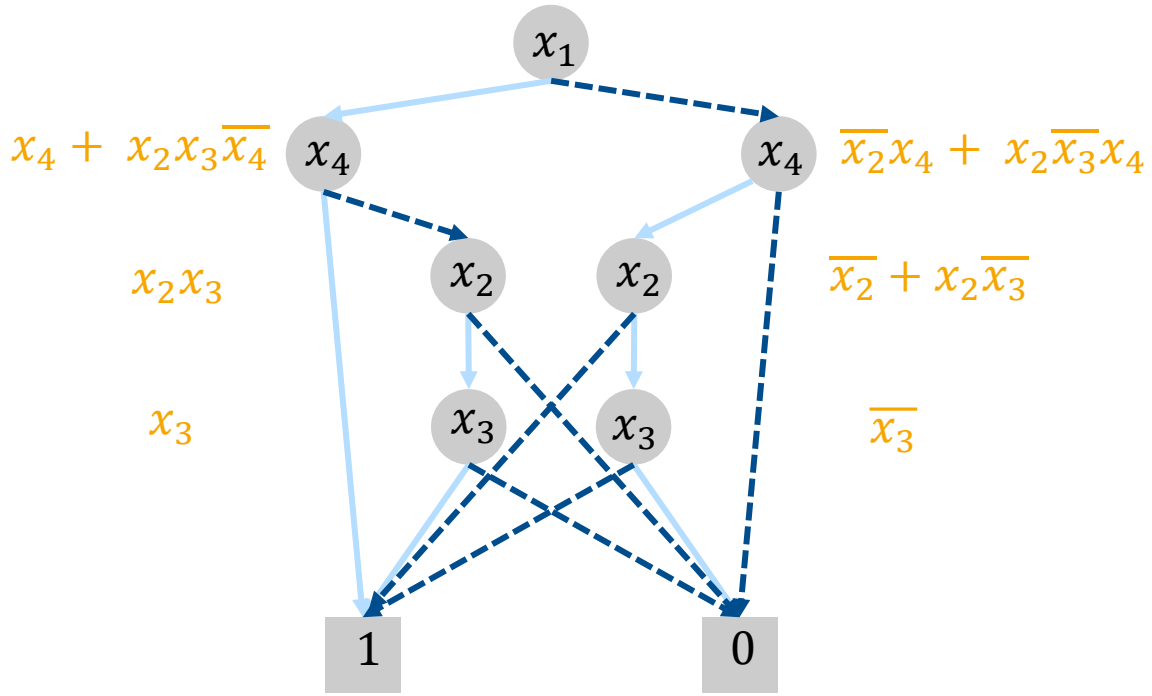
$f(x_1, x_2) = x_1 \oplus x_2$

Weitere Beispiele für BDDs

$x_1x_2x_3 + x_1x_2\overline{x_3}\,x_4 + x_1\overline{x_2}\;x_4 + \overline{x_1}\;\overline{x_2}\,x_4 + \overline{x_1}x_2\overline{x_3}\,x_4$



$x_1x_4 + x_1x_2x_3\overline{x_4} + \overline{x_1}\;\overline{x_2}x_4 + \overline{x_1}x_2\overline{x_3}x_4$



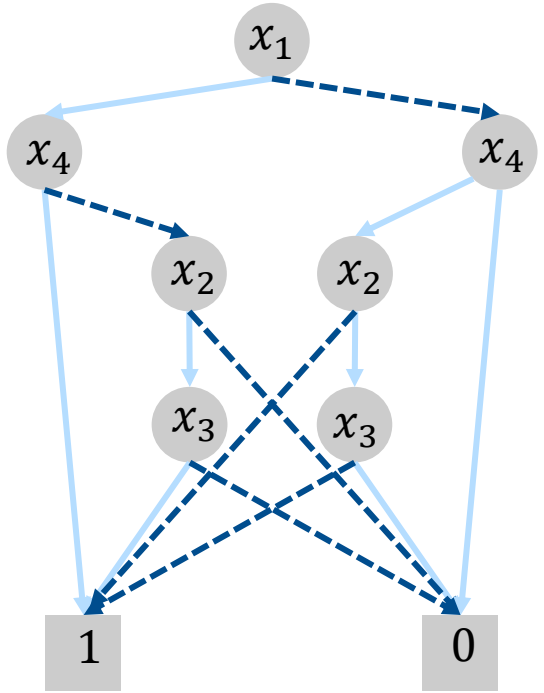
Beide beschreiben die Boolesche Funktion $x_1x_2x_3 + \overline{x_2}x_4 + \overline{x_3}x_4$ - doch keine eindeutige Darstellung?

Geordnete DDs

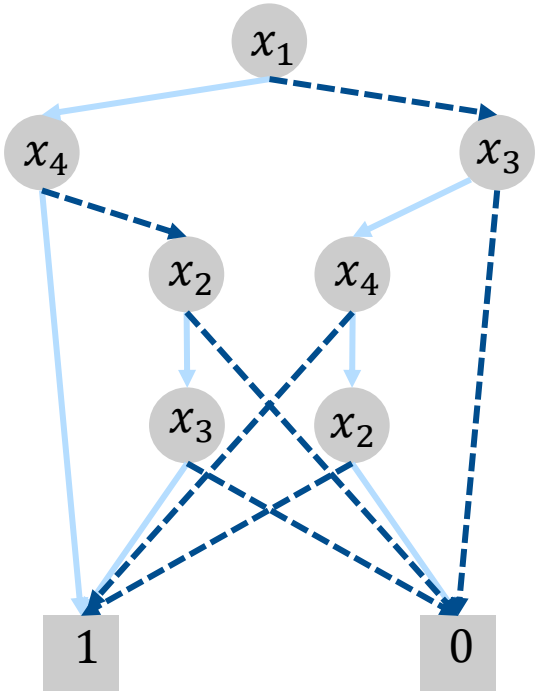
Definition (freies DD):
Ein Decision Diagram heißt frei, wenn auf jedem Pfad von der Wurzel zu einem Blatt jede Variable höchstens einmal als Label eines Knotens vorkommt.

Definition (geordnetes DD):
Ein Decision Diagram heißt geordnet, wenn auf jedem Pfad von der Wurzel zu einem Blatt die Variablen in der gleichen Reihenfolge abgefragt werden. Man schreibt auch OBDD für geordnete BDDs.

Beispiele (nicht) geordneter freier DDs



geordnetes freies DD

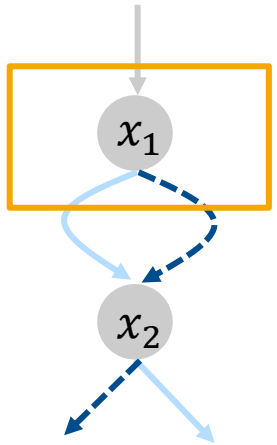


ungeordnetes freies DD

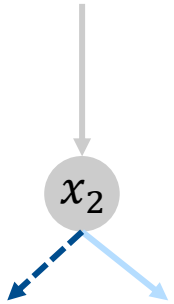
Redundante Knoten

Definition (redundanter Knoten im BDD):
Ein nicht-terminaler Knoten v mit $l(v) = x_i$ in einem Binary Decision Diagram heißt redundant, wenn $(f_v)_{x_i=0} = (f_v)_{x_i=1}$ gilt, das heißt beide Kofaktoren identisch sind.

Lemma:
Die Entfernung redundanter Knoten (und seiner Kind-Kanten) aus einem BDD ändert die dargestellte Boolesche Funktion nicht.



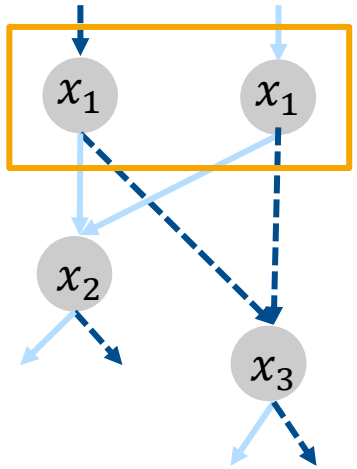
$$(f_{a_0})_{x_1=0} = f_{a_1} = (f_{a_0})_{x_1=1}$$



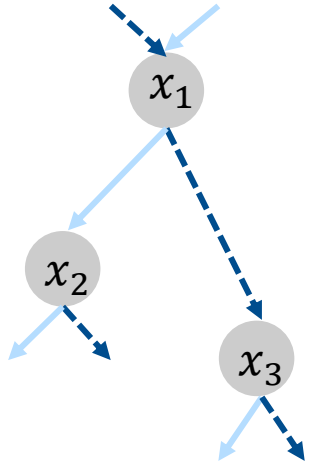
Isomorphe Knoten

Definition (isomorphe Knoten im BDD):
Zwei verschiedene Knoten v, w mit $l(v) = l(w) = x_i$ in einem Binary Decision Diagram heißen isomorph, wenn $low(v) = low(w)$ und $high(v) = high(w)$, das heißt, beide Kinder jeweils identisch sind.

Lemma:
Die Zusammenfassung isomorpher Knoten (und ihrer Kind-Kanten) in einem BDD ändert die dargestellte Boolesche Funktion nicht.



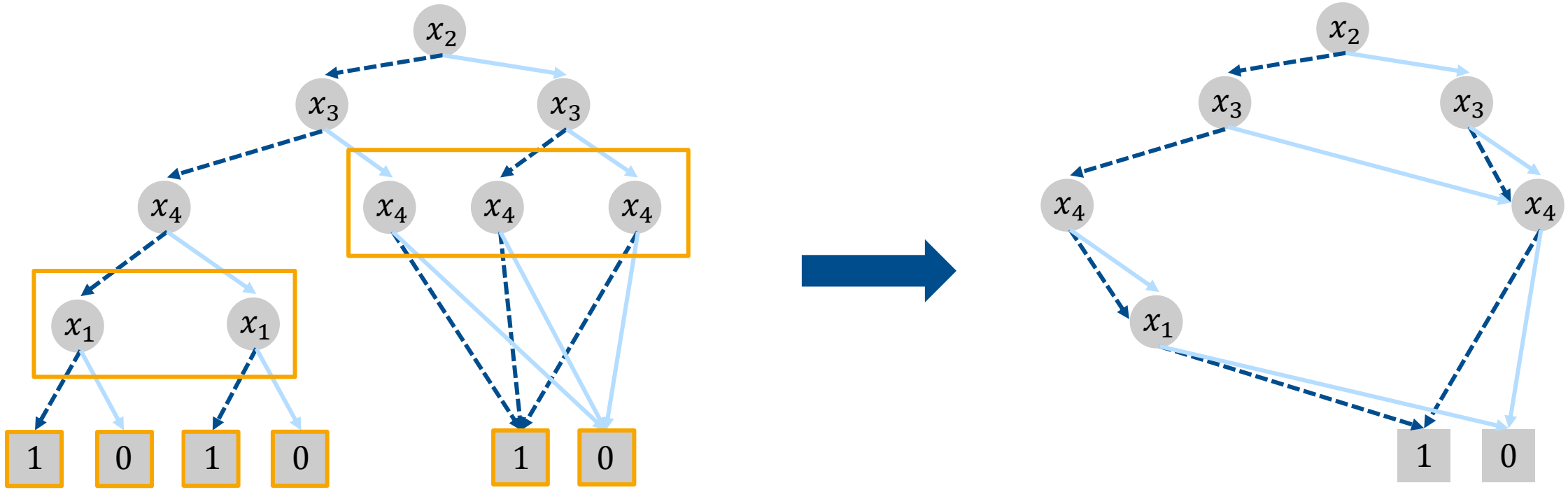
$$low(a_1) = a_4 = low(a_2)$$
$$high(a_1) = a_3 = high(a_2)$$



Reduzierte BDDs

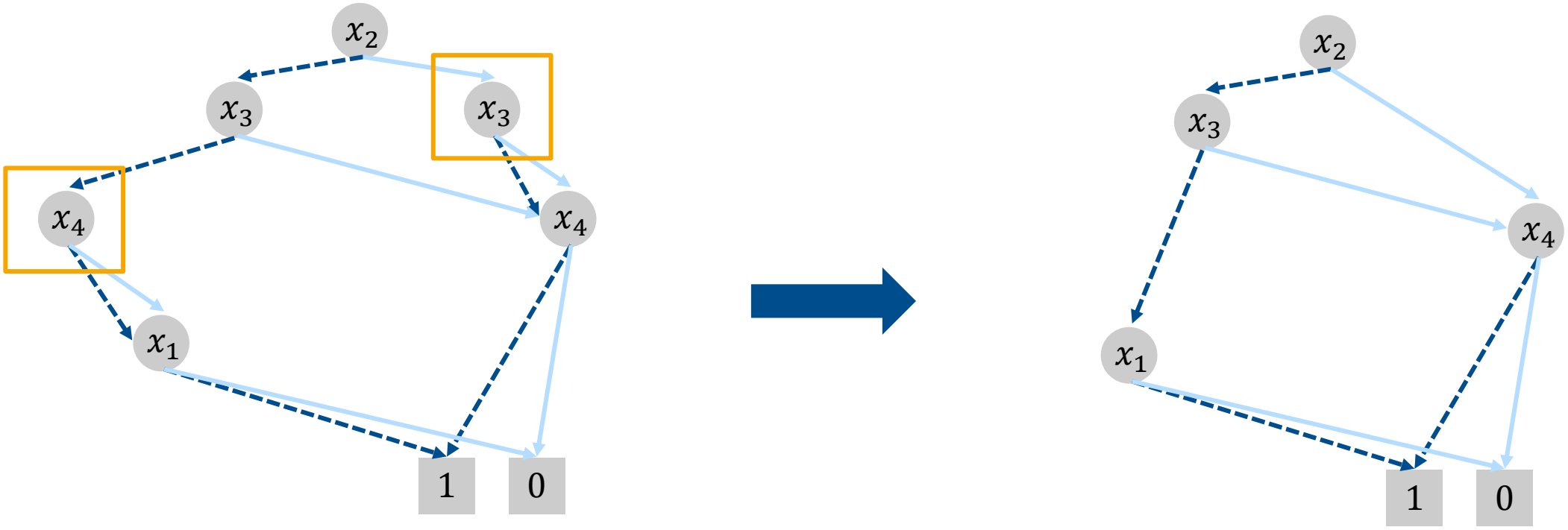
Definition (reduziertes BDD):
Ein Binary Decision Diagram heißt reduziert, wenn es keine isomorphen und keine redundanten Knoten enthält. Ein geordnetes reduziertes BDD nennt man auch ROBDD.

Beispiel für reduzierte und nicht-reduzierte BDDs



Isomorphe Knoten

Beispiel für reduzierte und nicht reduzierte Knoten - Fortsetzung



Redundante Knoten

Satz von Bryant (1986)

Satz von Bryant:
Geordnete reduzierte BDDs sind kanonische Darstellungen Boolescher Funktionen.

Schreibkonvention (im Folgenden):

BDD = reduziertes geordnetes BDD

Satz von Bryant: der Beweis (1)

Sei f in n Variablen über X_n gegeben.
Die Ordnung der Variablen sei oBdA $\pi = x_1 < x_2 \ldots < x_n$.

Wir zeigen, dass das BDD eindeutig bestimmt ist durch Induktion nach der Anzahl der Variablen, von denen f abhängt.

Wir zeigen zuerst Eindeutigkeit der BDD-Darstellung der konstanten Funktion **0** (d.h. $f(x_1, x_2, \ldots, x_n) = 0$).

Satz von Bryant: der Beweis (2) – Konstante Funktion

Sei also G ein BDD der konstanten Booleschen Funktion **0**. Dann gilt:

- Alle Pfade, die an der Wurzel von G starten, enden in dem mit 0 markierten Blatt.
- G enthält nur Blätter, die mit 0 markiert sind.
- Da G reduziert ist, enthält G nur genau ein mit 0 markiertes Blatt.
- Gäbe es in G einen inneren Knoten,
  - so gibt es auch einen inneren Knoten v, dessen Nachfolger beide Blätter sind, da G ein endlicher azyklischer Graph ist;
  - da G aber nur ein Blatt enthält, muss dieser Knoten v redundant sein.

Also:

G besteht genau aus dem mit 0 markierten Blatt und ist somit eindeutig .

Für die konstante Boolesche Funktion **1** läuft der Beweis analog.

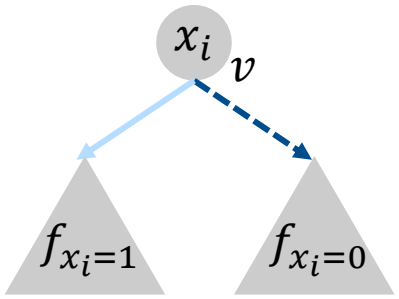
Satz von Bryant: der Beweis (3) - Induktionsschluss

Sei nun f eine Boolesche Funktion, die wenigstens von einer Variablen abhängt.

- Seien G_1 und G_2 zwei BDDs von f mit gleicher Variablenordnung.
- Die Wurzel von G_1 bzw. G_2 sei mit der Variablen x_i bzw.x_j markiert, d.h. es gilt $f_{G_1} = x_i f_{x_i=1}^{G_1} + \overline{x_i} f_{x_i=0}^{G_1}$ und $f_{G_2} = x_j f_{x_j=1}^{G_2} + \overline{x_j} f_{x_j=0}^{G_2}$.
- Nehme an, dass $x_i \neq x_j$ und dass oBdA $x_i < x_j$ (in Bezug auf π) ist, dann gilt:
  - f hängt von x_i ab, ansonsten wäre die Wurzel von G_1 redundant.
  - Kein Knoten von G_2 ist mit x_i markiert. G_2 beschreibt also eine Boolesche Funktion, die unabhängig von x_i ist. Widerspruch.
- Also gilt $x_i = x_j$.

Da die beiden Kofaktoren $f_{x_i=0}$ und $f_{x_i=1}$ Boolesche Funktionen sind, die von einer Variablen weniger abhängen als f, sind die dazugehörigen BDDs eindeutig bestimmt.

Hiermit folgt auch schon die Aussage.



Konstruktion eines BDDs über die Kofaktoren

„Top-Down“-Ansatz

Konstruiere BDD bdd_f von $f = x_1x_2x_3 + \overline{x_2}x_4 + \overline{x_3}x_4$ bzgl. der Variablenordnung $\pi = x_1 < x_2 < x_3 < x_4$

- Generiere sukzessive die BDDs für alle Kofaktoren
- Reduziere durch Entfernung redundanter Knoten und Zusammenfassung isomorpher Knoten

Konstruktion eines BDDs über Boolesche Operationen

„Bottom-Up“-Ansatz

Konstruiere BDD bdd_f von $f = x_1x_2x_3 + \overline{x_2}x_4 + \overline{x_3}x_4$

- 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 $bdd_{\overline{x_2}}$:=NOT(bdd_{x_2})
- Berechne $bdd_{\overline{x_2}\,x_4}$:=AND($bdd_{\overline{x_2}}, bdd_{x_4}$)
- Berechne $bdd_{\overline{x_3}}$:=NOT(bdd_{x_3})
- Berechne $bdd_{\overline{x_3}\,x_4}$:=AND($bdd_{\overline{x_2}}, bdd_{x_4}$)
- Berechne $bdd_{x_1x_2x_3+\overline{x_2}x_4}$:=OR($bdd_{x_1x_2x_3}, bdd_{\overline{x_2}\,x_4}$)
- Berechne bdd_f :=OR($bdd_{x_1x_2x_3+\overline{x_2}x_4}, bdd_{\overline{x_3}\,x_4}$)

- Zu realisieren sind also die binären Booleschen Operatoren

Manipulation von BDDs: Ideen

Definition (ITE-Operator):
Der ITE-Operator ist wie folgt definiert: $ITE(F,G,H) := FG + F\overline{H}$

Lemma:
Alle binären Booleschen Operatoren können auf den ternären Operator ITE zurückgeführt werden.

Beispiel:
$AND(F,G) = ITE\ (F,G,0)$
$OR(F,G) = ITE\ (F,1,G)$
$NOT(F) = ITE\ (F,0,1)$

Rückführung binärer Operatoren auf ITE

Funktion	Name	Ausdruck	ITE-Ausdruck
0000	0	0	0
0001	$AND(F,G)$	FG	$ITE(F,G,0)$
0010	$F > G$	$F\overline{G}$	$ITE(F,\overline{G},0)$
0011	F	F	F
0100	$F < G$	$\overline{F}G$	$ITE(F,0,G)$
0101	G	G	G
0110	$XOR(F,G)$	$F \oplus G$	$ITE(F,\overline{G},G)$
0111	$OR(F,G)$	$F + G$	$ITE(F,1,G)$

Funktion	Name	Ausdruck	ITE-Ausdruck
1000	$NOR(F,G)$	$\overline{F}\,\overline{G}$	$ITE(\overline{F},0,\overline{G})$
1001	$XNOR(F,G)$	$\overline{F \oplus G}$	$ITE(F,G,\overline{G})$
1010	$NOT(G)$	$\overline{G}$	$ITE(G,0,1)$
1011	$F \geq G$	$F + \overline{G}$	$ITE(F,1,\overline{G})$
1100	$NOT(F)$	$\overline{F}$	$ITE(F,0,1)$
1101	$F \leq G$	$\overline{F} + G$	$ITE(F,G,1)$
1110	$NAND(F,G)$	$\overline{F} + \overline{G}$	$ITE(F,\overline{G},1)$
1111	1	1	1

Idee zur Realisierung von ITE

Es gilt:

$$
\begin{aligned}
&ITE(F,G,H)\\
&= FG + \overline{F}H\\
&= x\left(FG + \overline{F}H\right)_{x=1} + \overline{x}\left(FG + \overline{F}H\right)_{x=0}\\
&= x\left(F_{x=1}G_{x=1} + \overline{(F_{x=1})}\,H_{x=1}\right) + \overline{x}\left(F_{x=0}G_{x=0} + \overline{(F_{x=0})}\,H_{x=0}\right)\\
&= x \cdot ITE(F_{x=1}, G_{x=1}, H_{x=1}) + \overline{x} \cdot ITE(F_{x=0,}\, G_{x=0}, H_{x=0})
\end{aligned}
$$

Vorgehen:

- Berechne rekursiv $ITE(F_{x=1}, G_{x=1}, H_{x=1})$ und $ITE(F_{x=0}, G_{x=0}, H_{x=0})$
- Überprüfe, ob ein Knoten v generiert wurde mit
  - $label(v) = x$
  - $f_{low(v)} = ITE(F_{x=0}, G_{x=0}, H_{x=0})$
  - $f_{high(v)} = ITE(F_{x=1}, G_{x=1}, H_{x=1})$
- Falls nicht, füge einen solchen Knoten ein.

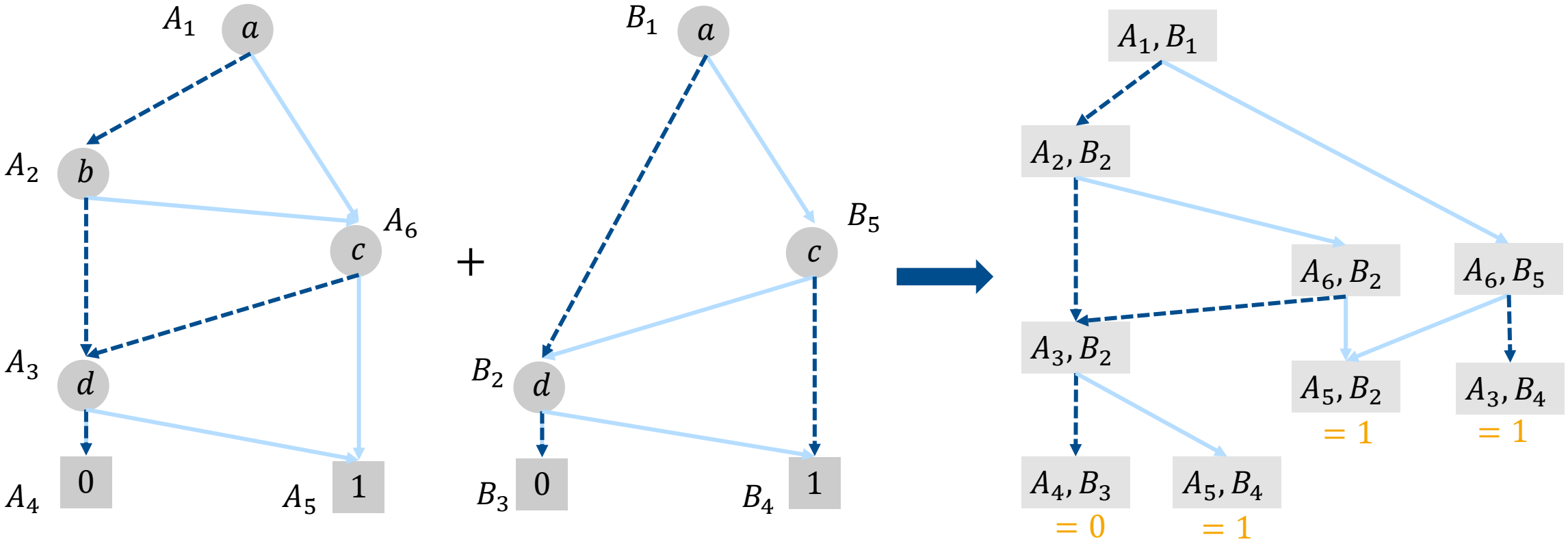
Zeitkomplexität: $\mathcal{O}(|F| \cdot |G| \cdot |H|)$

Realisierung von ITE

ITE(F,G,H) {
    if (terminal case **OR** (*F*,*G*,*H*) ∈ computed-table) {
        return result;
    } **else** {
        let x_i be the top variable of (*F*,*G*,*H*);
        $R_{high} = ITE(F^1_i, G^1_i, H^1_i)$;
        $R_{low} = ITE(F^0_i, G^0_i, H^0_i)$;
        if ($R_{high} = R_{low}$) {
            return R_{high};
        }
        $R = find\_or\_add\_unique\_table(v, R_{low}, R_{high})$;
        insert_computed_table(*F*,*G*,*H*,*R*);
        return *R*;
    }
}

BDD – Synthese (1)

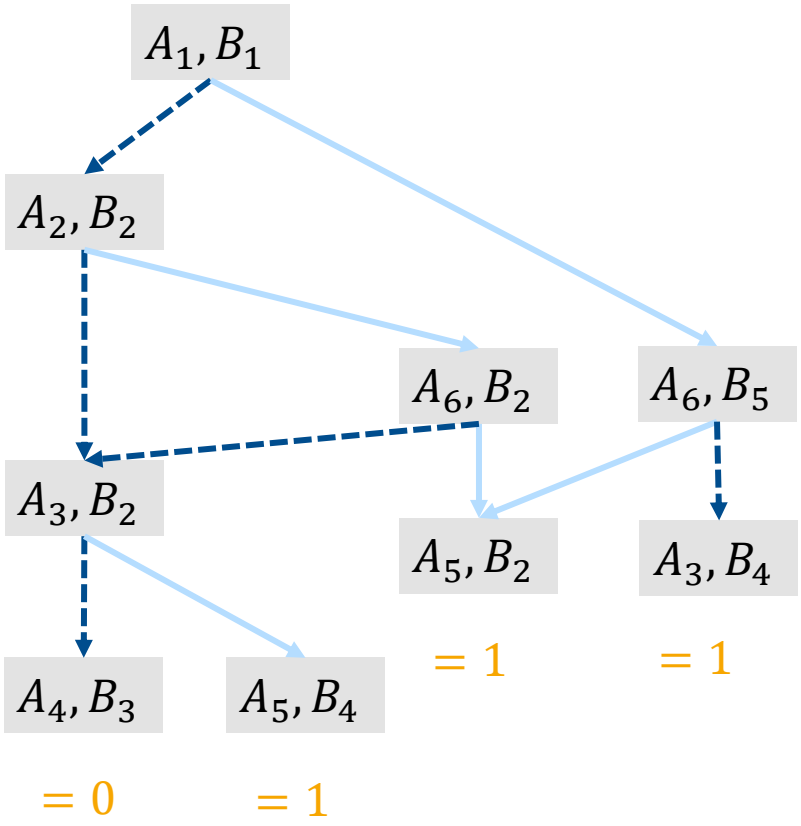
$F = ac + bc + d$ $\quad$ $G = a\bar{c} + d$ $\quad$ $F + G = ?$



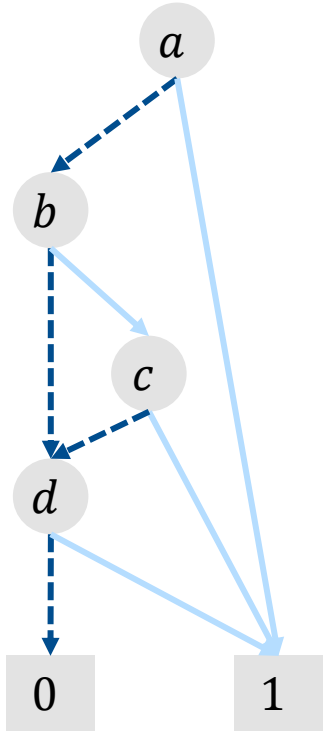
BDD Synthese (1)

$F = ac + bc + d$

$G = a\bar{c} + d$



$F + G = \partial a + bc + d$



Repräsentationsgröße

Kernproblem: gute Variablenordnung

Zur Erinnerung

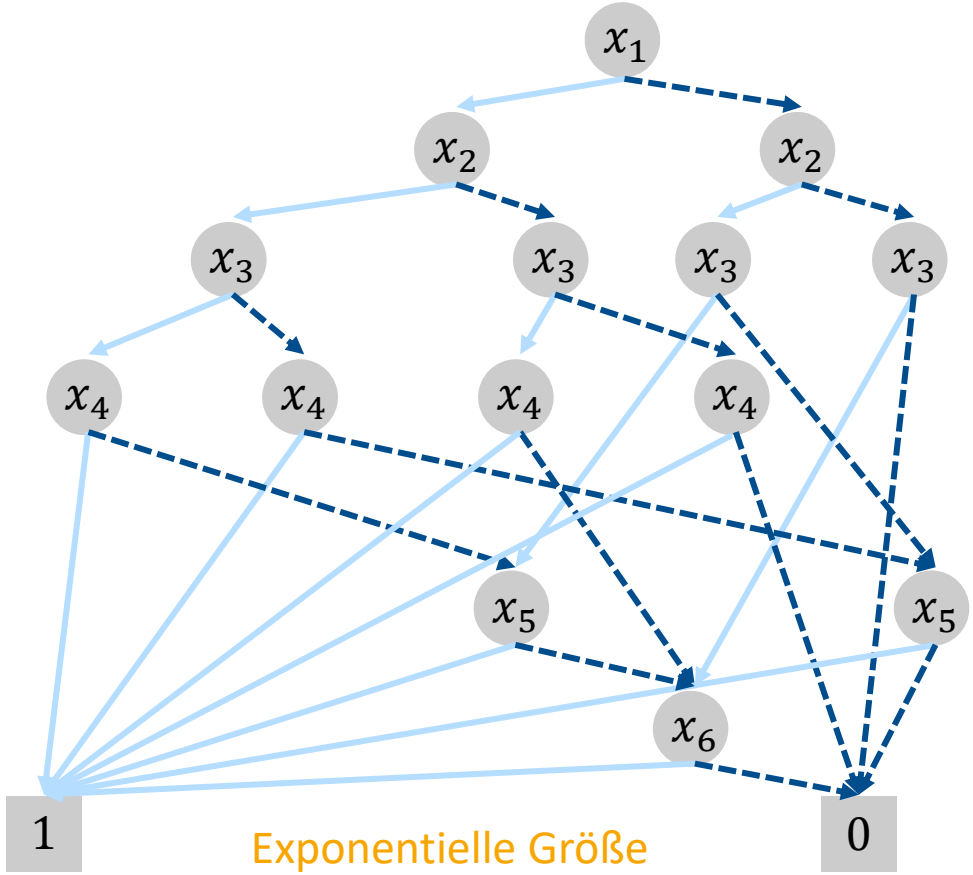
Für eine beliebige, aber feste Variablenordnung sind BDDs kanonische Darstellungen Boolescher Funktionen

Beobachtung
Die Größe der BDDs hängt sehr von der gewählten Variablenordnung ab !

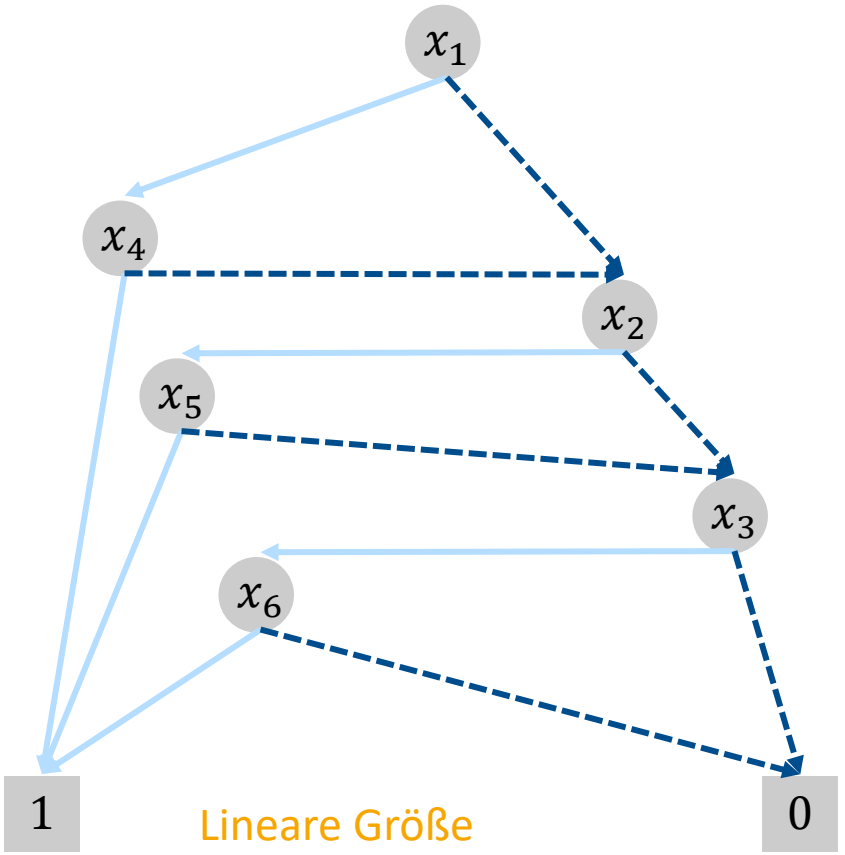
Beispiel: $f^{(n)}(x_1, \ldots, x_{2n}) = x_1 x_{n+1} + x_2 x_{n+2} + \cdots + x_n x_{2n}$

Einfluss der Variablenordnung: Beispiele

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



Exponentielle Größe



Lineare Größe

BDD-Minimierung

- Komplexitätstheoretisch schwierig
  - NP-vollständig
  - Es gibt (nur laufzeitintensive) exakte Algorithmen!

- Heuristiken zur BDD-Minimierung
  - **Initiale Verfahren** bauen ausgehend von einer anderen Darstellung einen ersten BDD auf
  - **Umordnungsverfahren** versuchen, eine bestehende Variablenordnung zu verbessern

- Die Multiplikation ist nur mit einem exponentiellen Aufwand in der Anzahl der Variablen (Bitbreite) durch BDDs darstellbar, unabhängig von der gewählten Ordnung.