Kapitel 4: Binäre Entscheidungsdiagramme (BDDs)

BDDs (binary decision diagrams)
wurden aus binären Entscheidungsbäumen für boole'sche Funktionen entwickelt.

Binärer Entscheidungsbaum

für Boole'sche Funktionen

(binary decision tree: BDT)

$f(a_1, a_2, b_1, b_2) = (a_1 \leftrightarrow b_1) \wedge (a_2 \leftrightarrow b_2)$

1 1 0 0

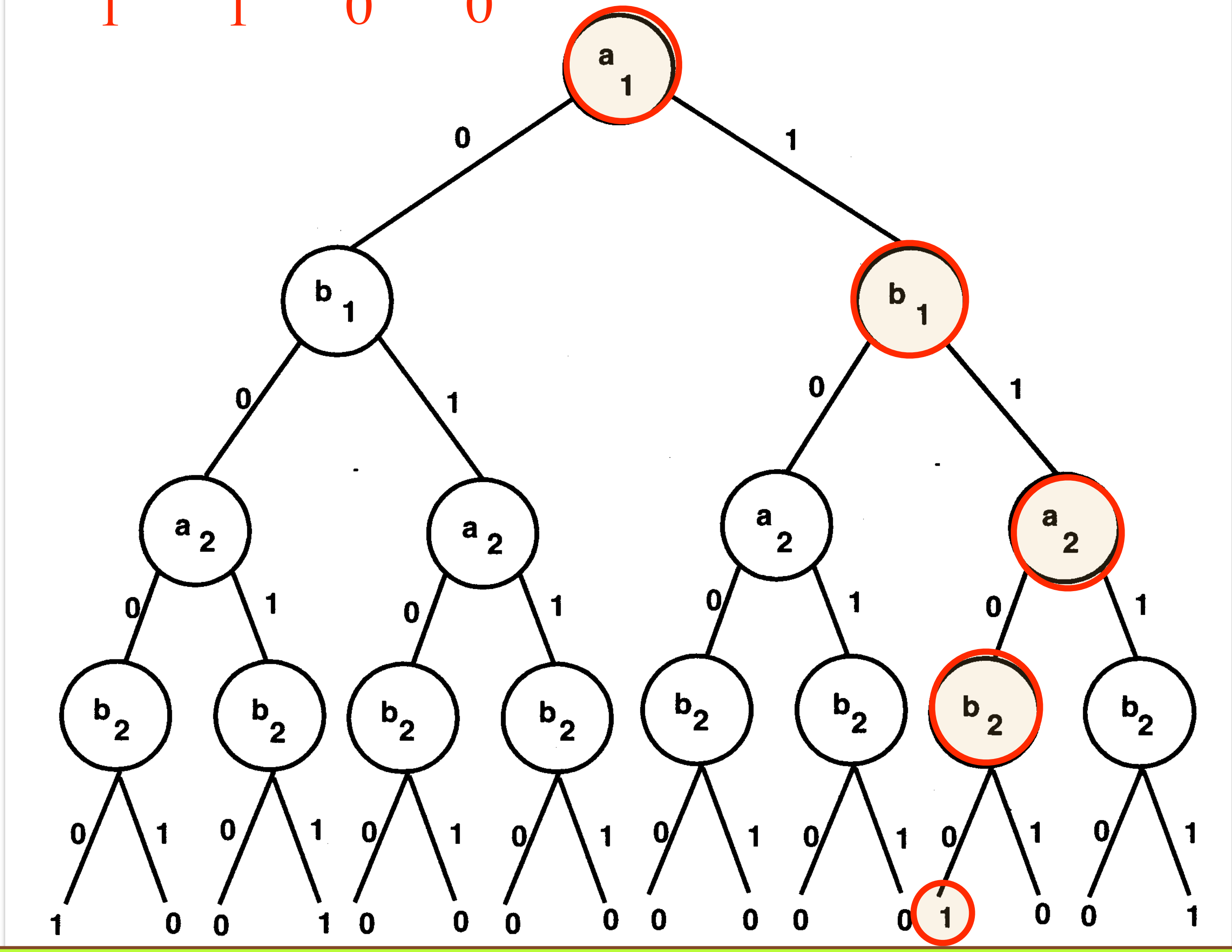


Abbildung 5.19: BDT für 2-bit Komparator

Größe von BDT's $\triangleq$ Größe von Entscheidungstabellen

$\longrightarrow$ exponentiell

aber viele redundante Informationen enthalten

$\longrightarrow$ Verschmelzen gleicher Unterbäume

Größe von BDT's $\hat{=}$ Größe von Entscheidungstabellen

$\longrightarrow$ exponentiell

aber viele redundante Informationen enthalten

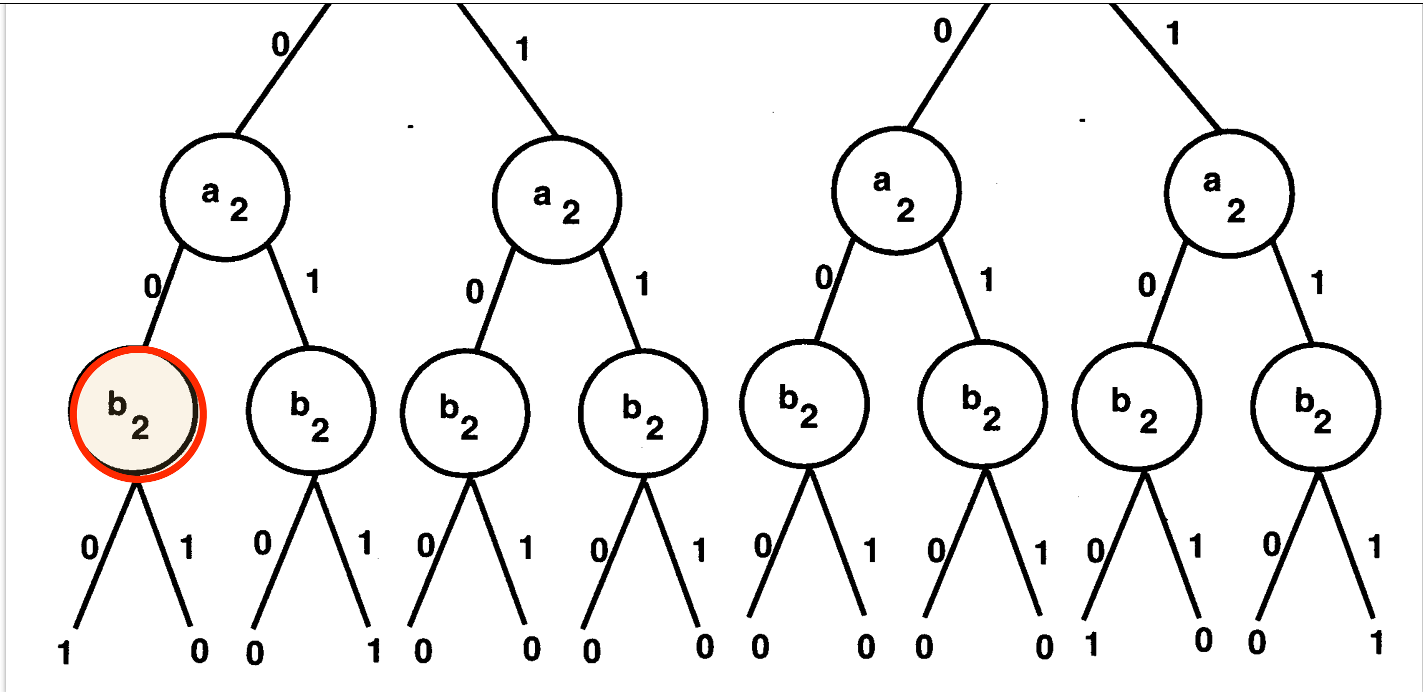
$\longrightarrow$ Verschmelzen gleicher Unterbäume

durch Funktion $f_v(x_1, \cdots, x_n)$ für einen Knoten v.

1. Wenn v ein Blatt ist, dann ist $f_v(x_1, \cdots, x_n) = value(v)$.

2. Wenn v kein Blatt ist, dann gilt für $x_i = var(v)$:

   $f_v(x_1, \cdots, x_n) = (\neg x_i \wedge f_{low(v)}(x_1, \cdots, x_n)) \vee (x_i \wedge f_{high(v)}(x_1, \cdots, x_n))$,



Beispiel 5.47 Für den inneren Knoten links unten in Abb. 5.19 gilt:

$$f_v(a_1, a_2, b_1, b_2) = (\neg b_2 \wedge 1) \vee (b_2 \wedge 0) = \neg b_2$$

5.8.1 Normalformen und logische Operationen auf BDD’s

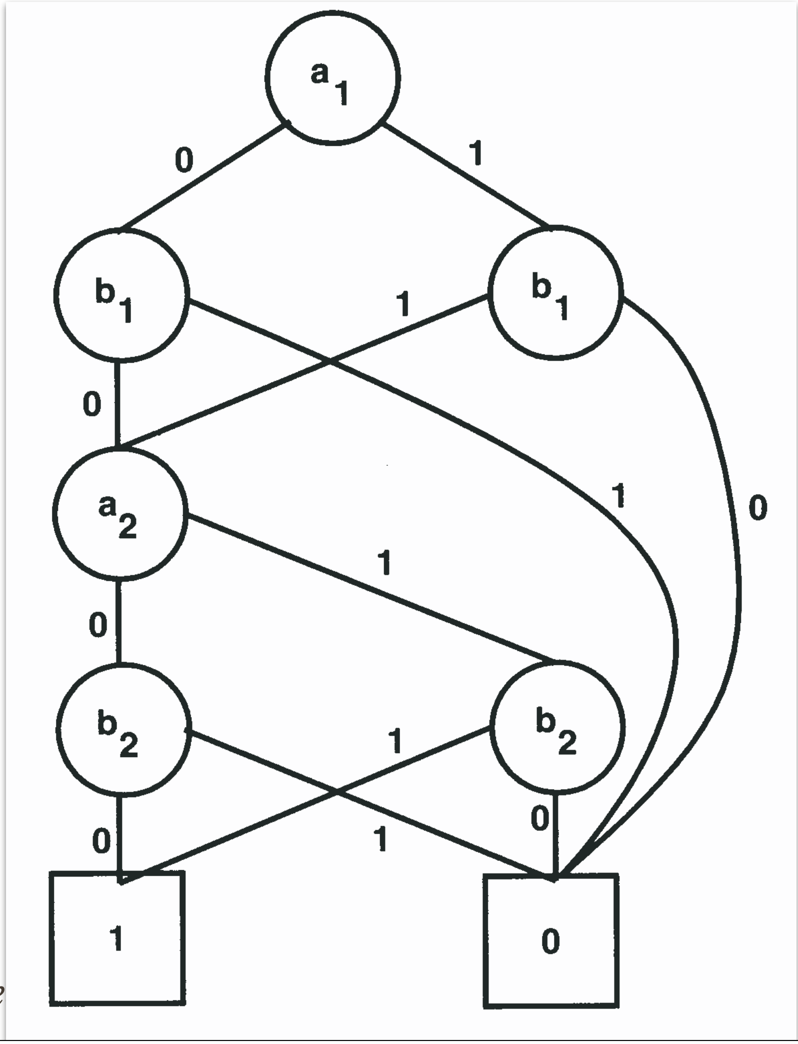
Wünschenswert ist eine Normalform für BDD’s. Notwendige Eigenschaft: Zwei BDD’s sind genau dann äquivalent, wenn für beide *isomorphe* Repräsentationen existieren. Dazu zwei Bedingungen:

1. Die Variablen müssen geordnet sein. Geordnete BDD’s heißen OBDD’s.

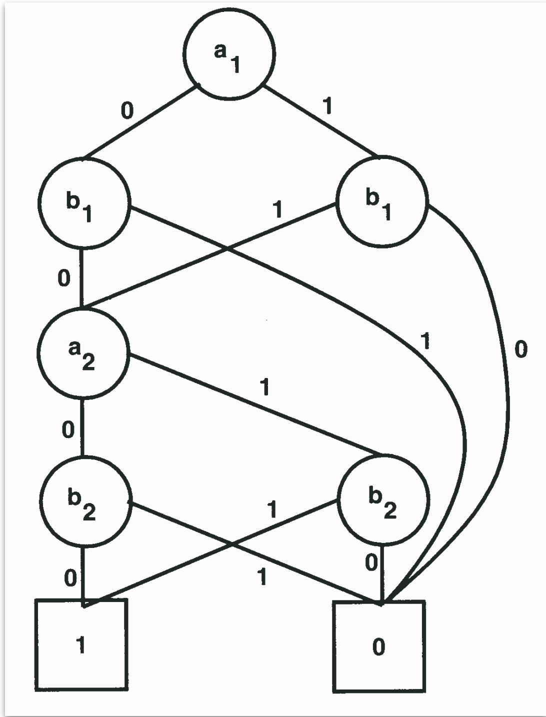
2. Keine redundante Unterbäume.

1. Die Variablen müssen geordnet sein. Geordnete BDD's heißen OBDD's.

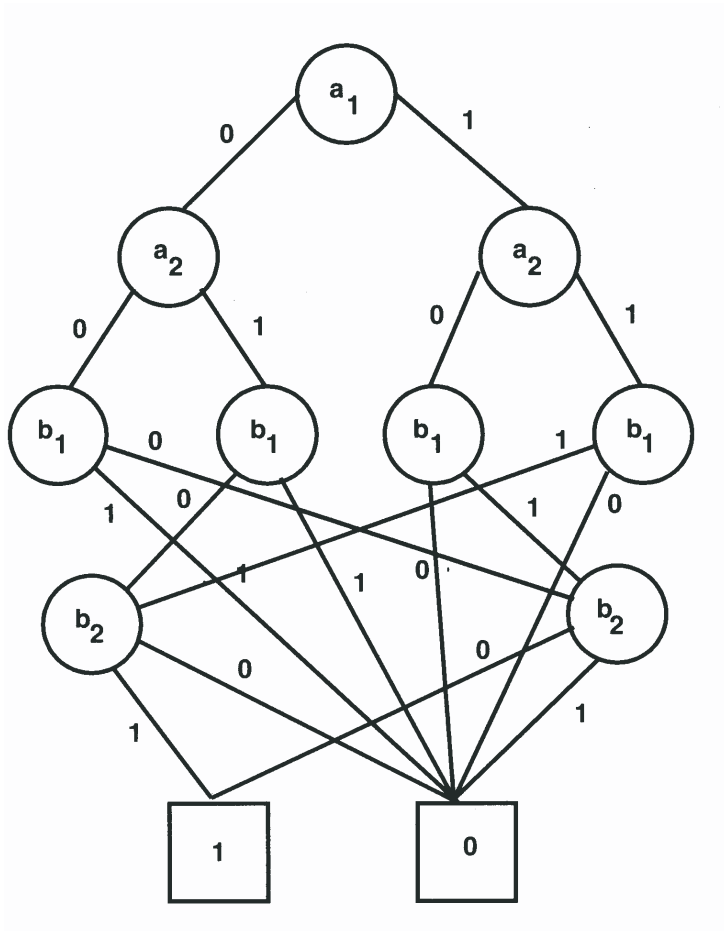
Es folgt das Ergebnis für das Beispiel 5.46 mit der Ordnung $a_1 < b_1 < a_2 < b_2$. Bei so geordneten Variablen erhält man im allgemeinen für einen n-bit Komparator $3n + 2$ Knoten.



1. Die Variablen müssen geordnet sein. Geordnete BDD's heißen OBDD's.



Mit der Ordnung $a_1 < a_2 < b_a < b_2$ erhält man in allgemeinem $3 \cdot 2^n - 1$ Knoten.



Das Problem, eine optimale Ordnung zu finden, ist NP-Vollständig!

Es gibt aber viele heuristische Ansätze, wie z.B. die Nähe von Gattern im Layout.

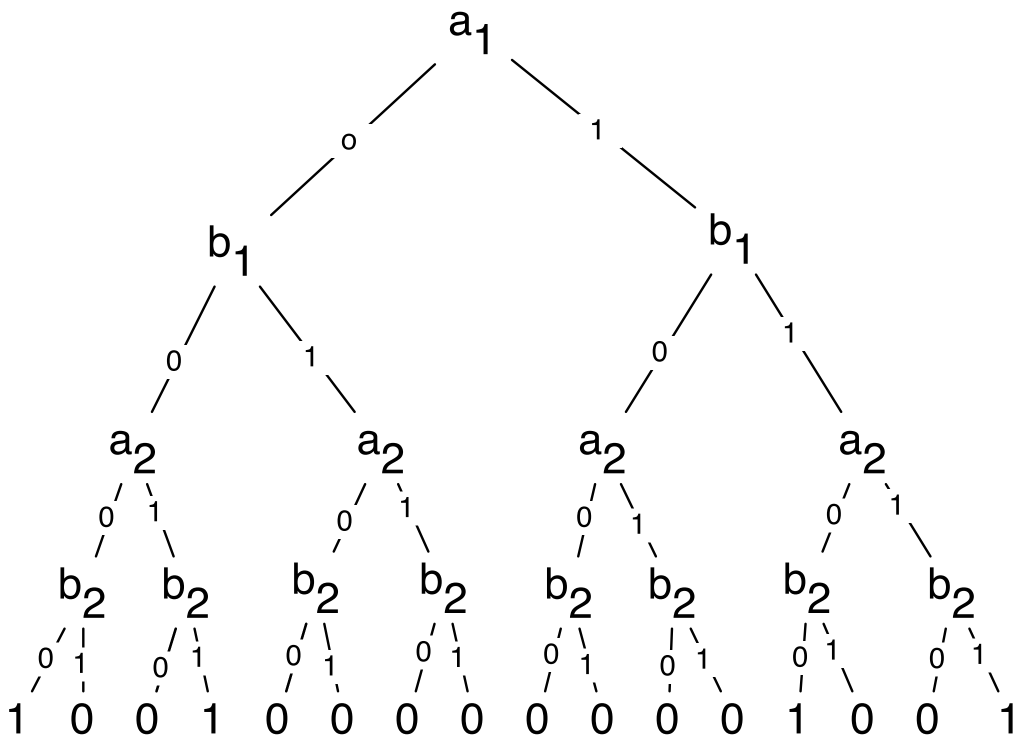
2. Keine redundante Unterbäume. Dazu folgende Prozedur *Reduce*:

    - *Entferne doppelte Blätter:* Entferne alle bis auf ein Blatt mit gleicher Auszeichnung und biege alle jetzt losen Kanten zu diesem Blatt.

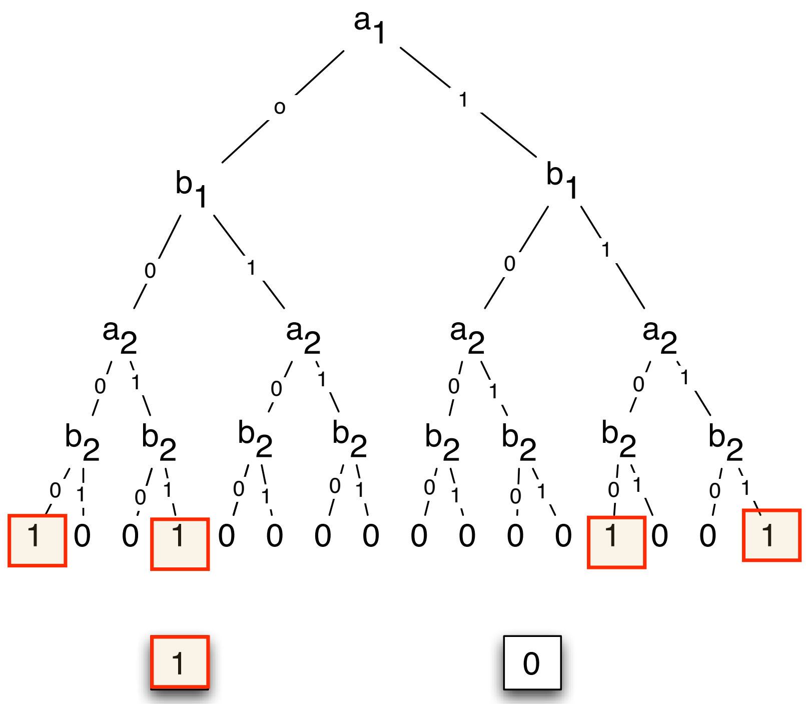
    - *Entferne doppelte innere Knoten:* Für zwei innere Knoten u und v:
      Wenn $var(u) = var(v)$, $low(u) = low(v)$ und $high(u) = high(v)$, dann lösche einen der Knoten und seine abgehenden Kanten. Die eingehenden Kanten werden zu dem anderen hingezogen.

- *Entferne redundante Abfragen:* Für einen inneren Knoten v:
  Wenn $low(v) = high(v)$, dann entferne v und seine abgehenden Kanten. Die eingehenden Kanten werden zu $low(v)$ hingezogen.

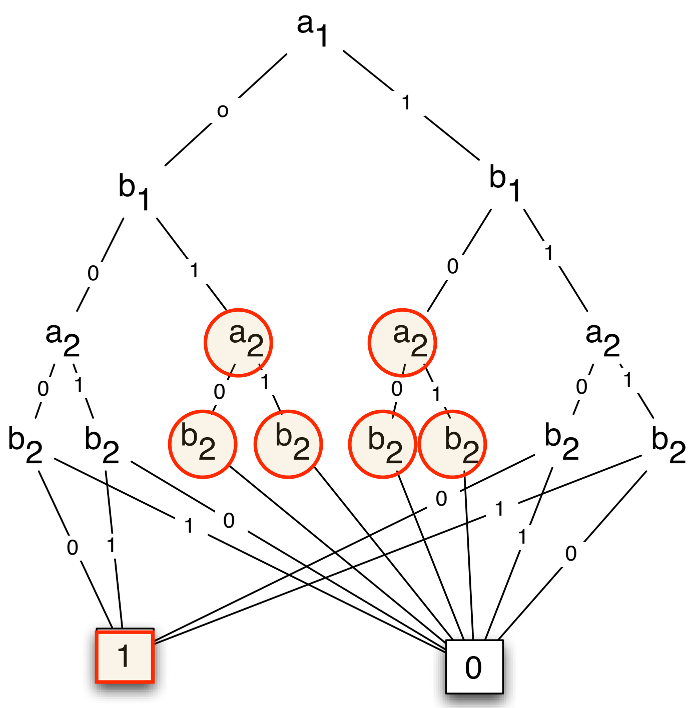
Beispiel: 2-bit-Komparator



- *Entferne doppelte Blätter:* Entferne alle bis auf ein Blatt mit gleicher Auszeichnung und biege alle jetzt losen Kanten zu diesem Blatt.

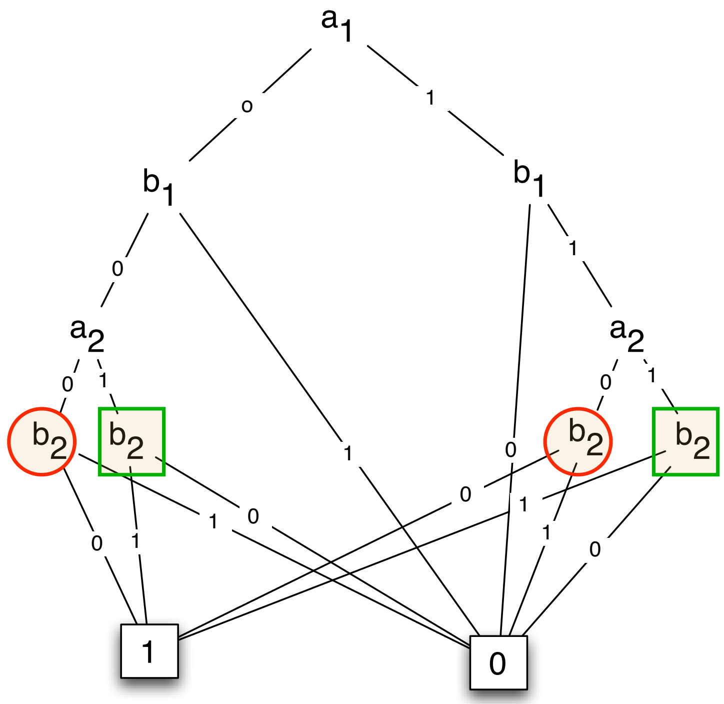


- *Entferne doppelte innere Knoten:* Für zwei innere Knoten u und v:
  Wenn $var(u) = var(v)$, $low(u) = low(v)$ und $high(u) = high(v)$, dann lösche einen der Knoten und seine abgehenden Kanten. Die eingehenden Kanten werden zu dem anderen hingezogen.
- *Entferne redundante Abfragen:* Für einen inneren Knoten v:
  Wenn $low(v) = high(v)$, dann entferne v und seine abgehenden Kanten. Die eingehenden Kanten werden zu $low(v)$ hingezogen.



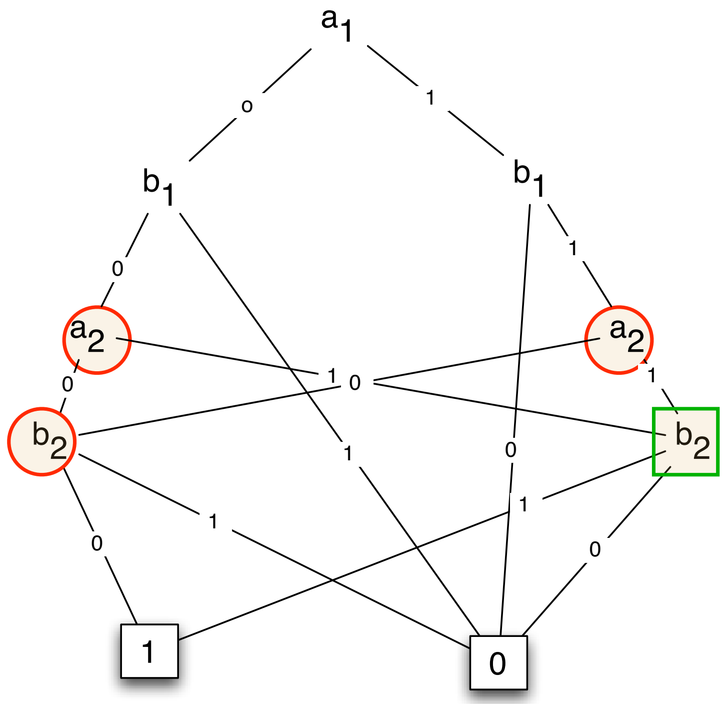
- *Entferne doppelte innere Knoten:* Für zwei innere Knoten u und v:
  Wenn $var(u) = var(v)$, $low(u) = low(v)$ und $high(u) = high(v)$, dann lösche einen der Knoten und seine abgehenden Kanten. Die eingehenden Kanten werden zu dem anderen hingezogen.

- *Entferne redundante Abfragen:* Für einen inneren Knoten v:
  Wenn $low(v) = high(v)$, dann entferne v und seine abgehenden Kanten. Die eingehenden Kanten werden zu $low(v)$ hingezogen.



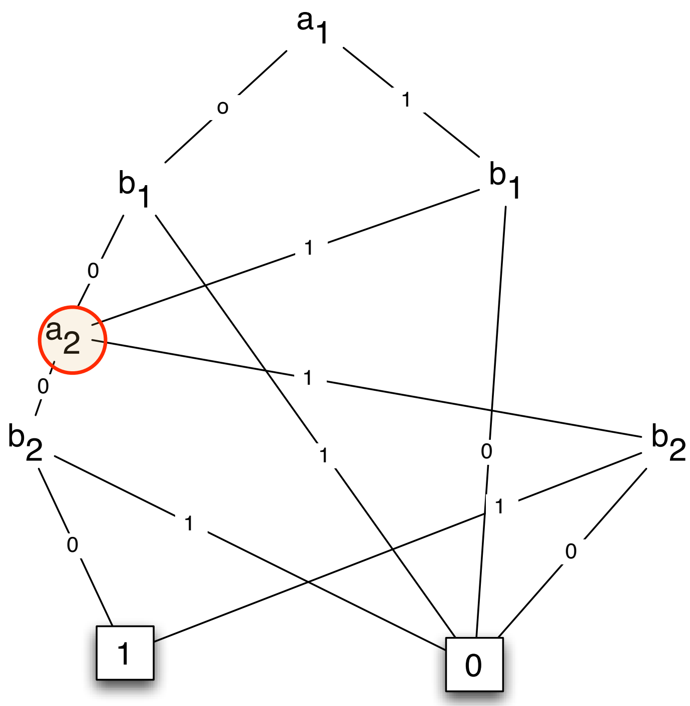
- *Entferne doppelte innere Knoten:* Für zwei innere Knoten u und v:
  Wenn $var(u) = var(v)$, $low(u) = low(v)$ und $high(u) = high(v)$, dann lösche einen der Knoten und seine abgehenden Kanten. Die eingehenden Kanten werden zu dem anderen hingezogen.

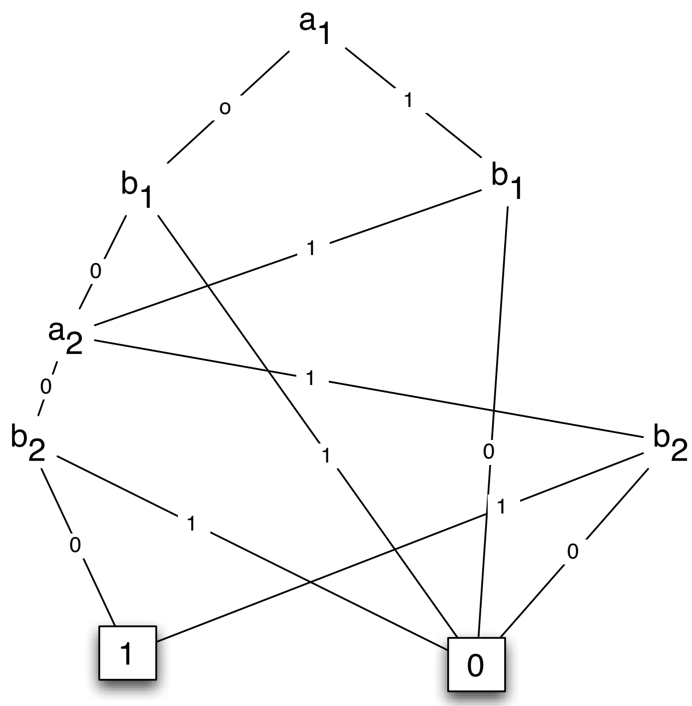
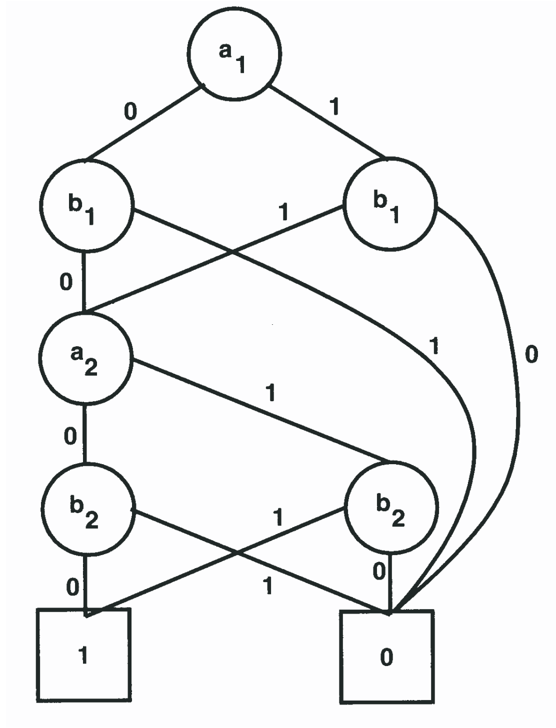
- *Entferne redundante Abfragen:* Für einen inneren Knoten v:
  Wenn $low(v) = high(v)$, dann entferne v und seine abgehenden Kanten. Die eingehenden Kanten werden zu $low(v)$ hingezogen.



- *Entferne doppelte innere Knoten:* Für zwei innere Knoten u und v:
  Wenn $var(u) = var(v)$, $low(u) = low(v)$ und $high(u) = high(v)$, dann lösche einen der Knoten und seine abgehenden Kanten. Die eingehenden Kanten werden zu dem anderen hingezogen.

- *Entferne redundante Abfragen:* Für einen inneren Knoten v:
  Wenn $low(v) = high(v)$, dann entferne v und seine abgehenden Kanten. Die eingehenden Kanten werden zu $low(v)$ hingezogen.



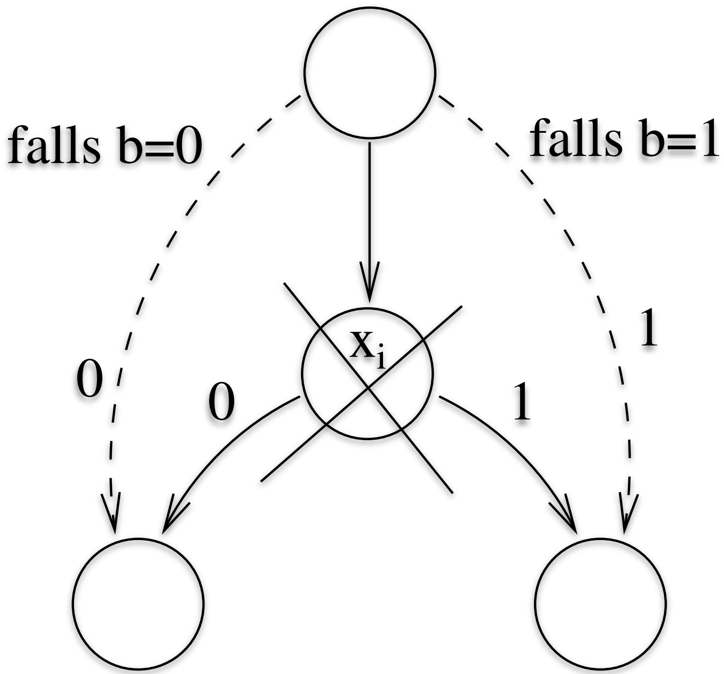

a1
o
1
b1
b1
0
1
a2
1
0
b2
b2
1
0
1
0
1
0
1
0

a1
0
1
b1
b1
1
0
a2
1
0
1
b2
b2
1
0
0
1
1
0

Logische Operationen auf BDD's:

a) Einschränken eines Argumentes x_i der Funktion f auf einen Wert $b \in \{0,1\}$

$$f_{|x_i \leftarrow b}(x_1, \cdots, x_n) = f(x_1, \cdots, x_{i-1}, b, x_{i+1}, \cdots, x_n)$$

OBDD von $f_{|x_i \leftarrow b}$ aus OBDD von f:

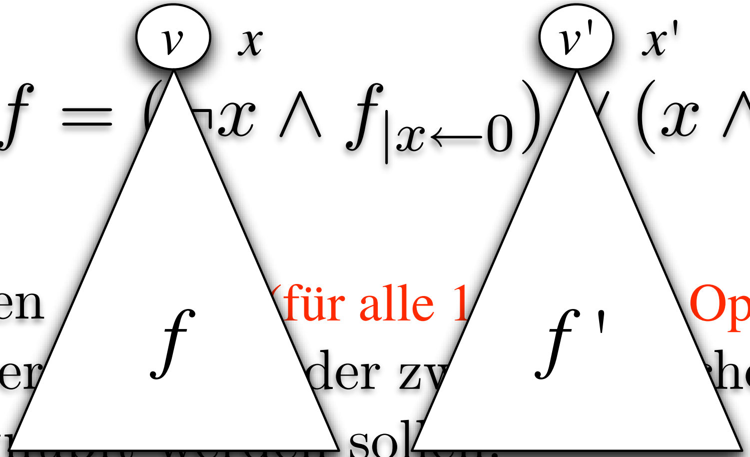


Shanon-Expansion

$$f = (\neg x \wedge f_{|x \leftarrow 0}) \vee (x \wedge f_{|x \leftarrow 1})$$

Allgemeines Verfahren *Apply*: (für alle 16 binären Operatoren)
Sei $\star$ die logische Operation, mit der zwei boole'schen Funktionen f und f' verknüpft werden sollen.

Shanon-Expansion



$$f = (\neg x \wedge f_{|x \leftarrow 0}) \vee (x \wedge f_{|x \leftarrow 1})$$

Allgemeines Verfahren … (für alle 1… Operatoren)
Sei $\star$ die logische Oper… der zw…hen Funktionen f und f' verk… werden sollen.

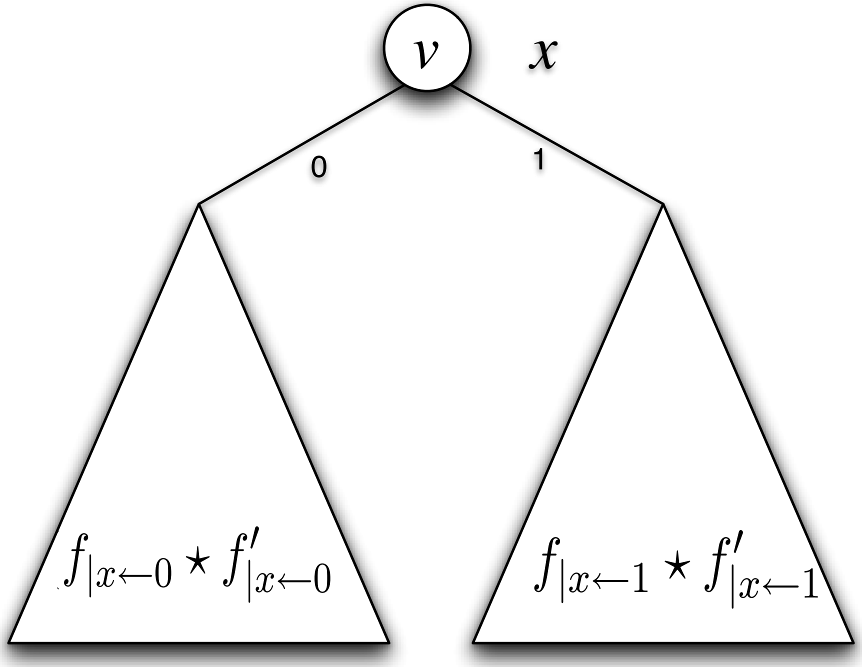
- v und v' sind die Wurzel-Knoten von OBDD's f und f',

- Wenn v und v' Blätter sind, dann $f \star f' = value(v) \star value(v')$, sonst

- Sei $x = var(v)$ und $x' = var(v')$:

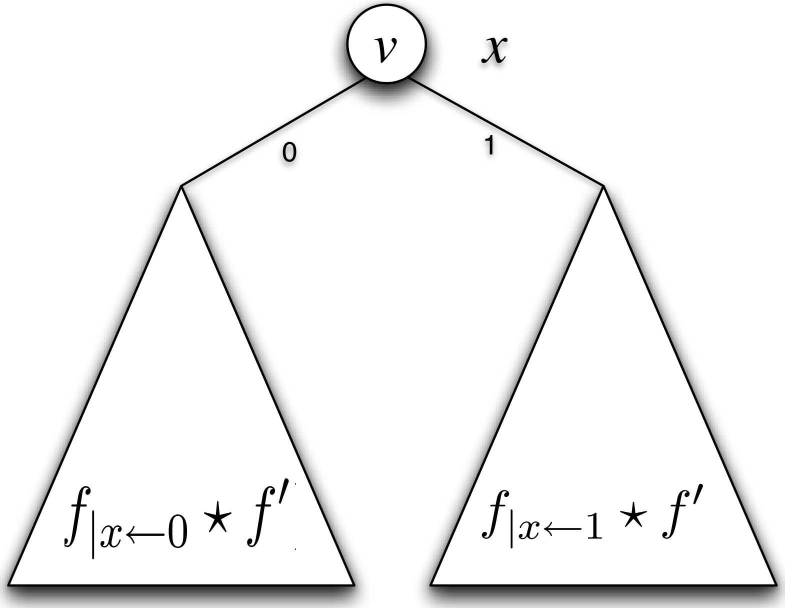
a) Wenn $x = x'$, dann „Shanon-Expansion”:

$$f \star f' = (\neg x \wedge (f_{|x \leftarrow 0} \star f'_{|x \leftarrow 0})) \vee (x \wedge (f_{|x \leftarrow 1} \star f'_{|x \leftarrow 1}))$$



b) Wenn $x < x'$, dann ist f' unabhängig von x. Also:

$$f \star f' = (\neg x \wedge (f_{|x \leftarrow 0} \star f')) \vee (x \wedge (f_{|x \leftarrow 1} \star f'))$$



c) Wenn $x' < x$, dann ist f unabhängig von x'. Also:

$$f \star f' = (\neg x' \wedge (f \star f'_{|x' \leftarrow 0})) \vee (x' \wedge (f \star f'_{|x' \leftarrow 0}))$$

- OBDD : Bryant 1986
- symbolische Darstellung von Transitionssystemen : McMillan 1987
- Anwendung des CTL-Algorithmus von Clarke/Emerson 1994 erlaubt drastische Erhöhung der Größe der Zustandsmenge, z.B. 10^{20} Zustände
- Model Checking System von McMillan : SMV
- Beispiel einer realen Anwendung:
  IEEE futurebus standard von 1988, Verfikation erst 1992 mit SMV

5.8.2 Darstellung von Kripke-Strukturen durch OBDD's

Für Relation $Q \subseteq \{0,1\}^n$ mit charakteristischer Funktion:

$$f_Q(x_1, \cdots, x_n) = 1 \Leftrightarrow Q(x_1, \cdots, x_n)$$

Falls $Q \subseteq D^n$ mit $|D| = 2^m, \quad m > 1$ wird eine Bijektion ϕ definiert:

$$\phi : \{0,1\}^m \to D$$

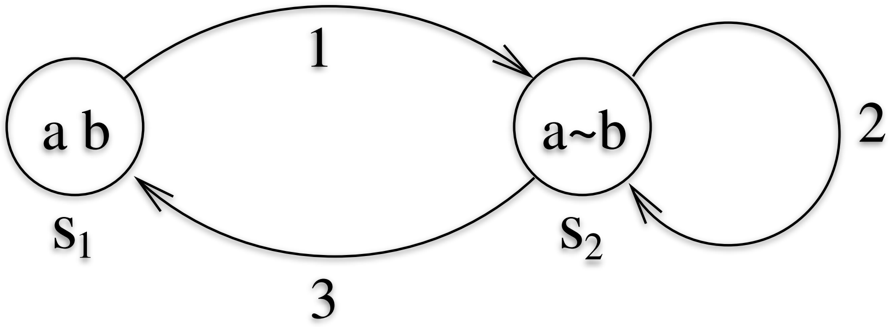
Daraus neue Relation $\hat{Q} \subseteq \{0,1\}^{m \cdot n}$:

$$\hat{Q}\ (\bar{x}_1, \cdots, \bar{x}_n) = Q(\phi(\bar{x}_1), \cdots, \phi(\bar{x}_n))$$

Die Kripke-Struktur $M = (S, R, L)$ wird folgendermaßend kodiert:

- Zustandsmenge S mit $\phi : \{0,1\}^m \to S$
- Transitionsreleation R mit $\hat{R}\,(\bar{x}, \bar{x}') = R(\phi(\bar{x}), \phi(\bar{x}'))$
- Auszeichnungsmengen $L : L_p := \{s \,|\, p \in L(s)\}$

Beispiel 5.48



Dazu die Formel:

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

wobei a, b die Ausgang- und a', b' Nachfolgerzustandsvariablen sind.