Grundlagen digitaler Systeme WS12

Binary Decision Diagrams

Johann Blieberger

183.580, VU 2.0

Automation Systems Group E183-1
Institute of Computer Aided Automation
Vienna University of Technology
email: gds@auto.tuwien.ac.at

Darstellungsmöglichkeiten kombinatorischer Logik

- Disjunktive-/konjunktive Normalform
  - Geeignet zur Implementierung in Hardware (Gatterschaltungen, PLA, etc.)
- Darstellung mittels If-Then-Else (ITE)
  - Z.B.:

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

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

Wahrheitstabelle und ITE

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

Shannon-Zerlegung

- Sei $\mathbf{f} : \mathbf{B}^n \rightarrow \mathbf{B}$ eine Boolesche Funktion.
- Definiere **Kofaktor** von **f** nach $x_i = 1$ durch $\mathbf{f_{i,1}} : \mathbf{B}^n \rightarrow \mathbf{B}$

  $$\mathbf{f}_{i,1}(a_1,\ldots,a_{i-1}, a_i, a_{i+1},\ldots,a_n) = \mathbf{f}(a_1,\ldots,a_{i-1}, 1, a_{i+1},\ldots,a_n)$$

  für alle $a_k \in \{0,1\}$, $k=1,\ldots,n$.
- Entsprechend heißt $\mathbf{f_{i,0}}$ **Kofaktor** von f nach $x_i = 0$.
- Es gilt dann $\mathbf{f} = (x_i \wedge \mathbf{f}_{i,1}) \vee (\neg x_i \wedge \mathbf{f}_{i,0})$.
- $\mathbf{f} = (x_i \wedge \mathbf{f}_{i,1}) \vee (\neg x_i \wedge \mathbf{f}_{i,0})$ heißt **Shannon-Zerlegung** von f.

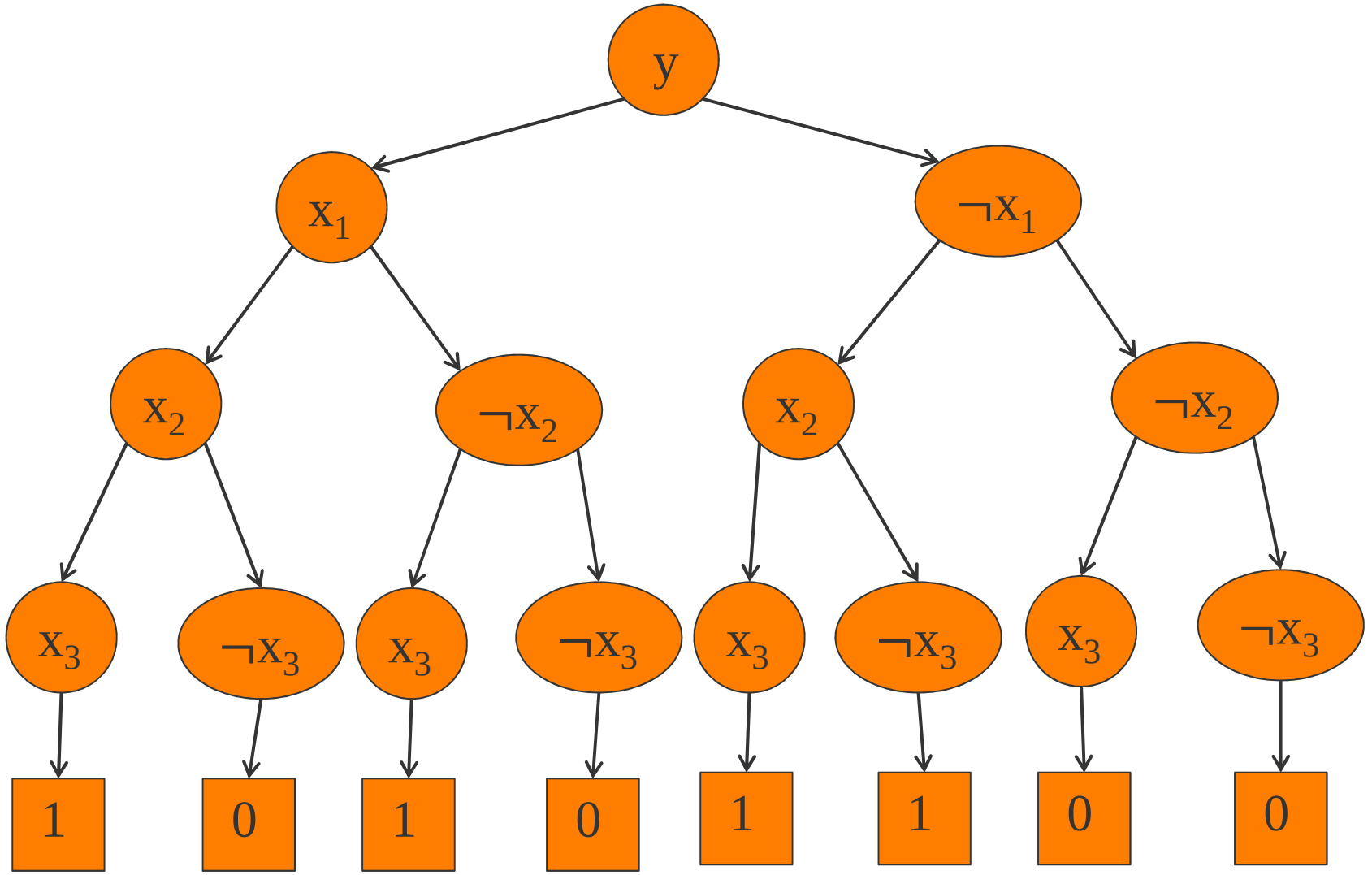
Wahrheitstabelle und ITE

Beispiel

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

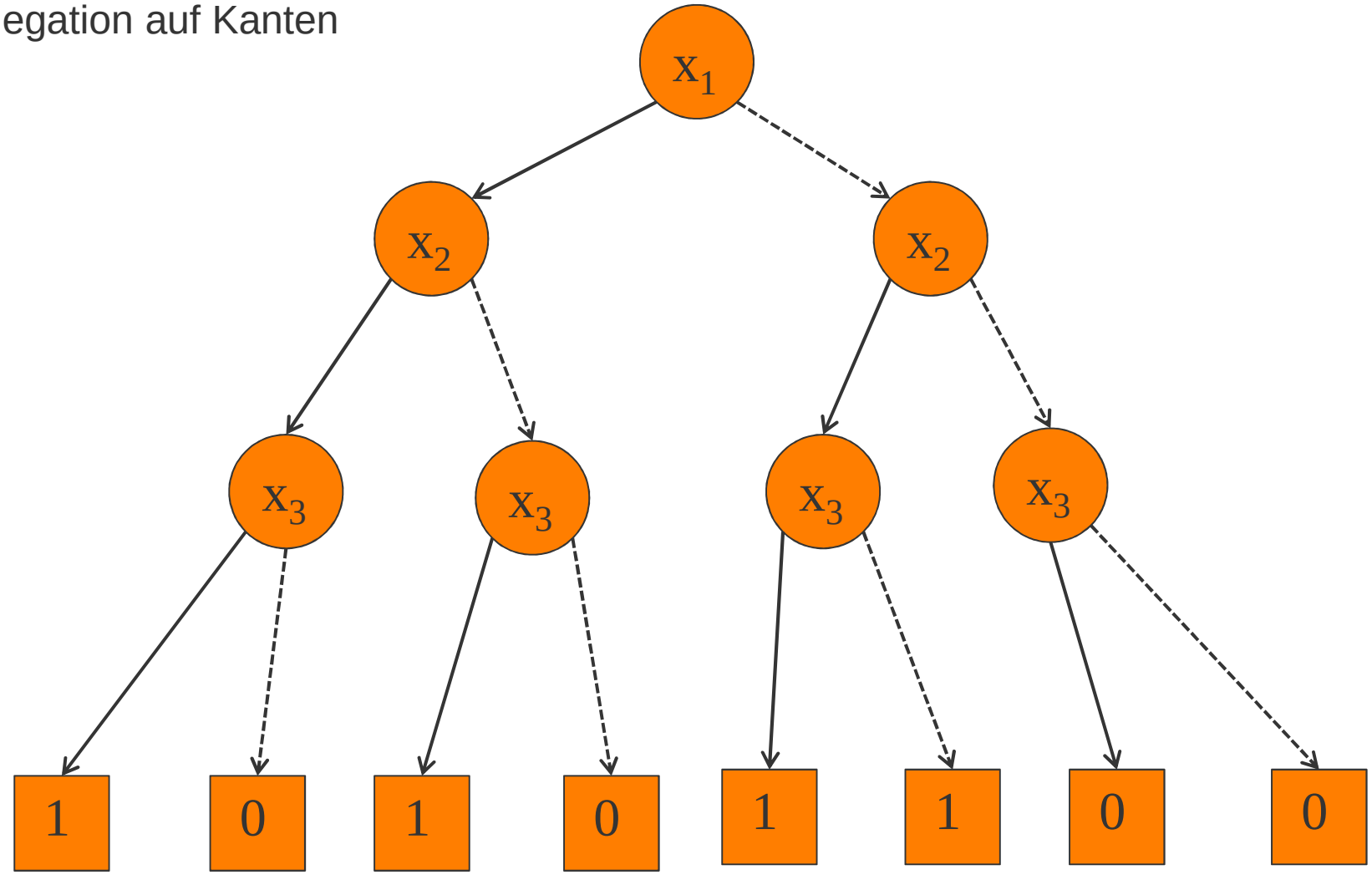
```
IF x1 THEN
        IF x2 THEN
                IF x3 THEN 1
                ELSE 0
        ELSE
                IF x3 THEN 1
                ELSE 0
ELSE
        IF x2 THEN
                IF x3 THEN 1
                ELSE 1
        ELSE
                IF x3 THEN 0
                ELSE 0
```

Decision Tree



Decision Tree: Andere Schreibweise

Negation auf Kanten



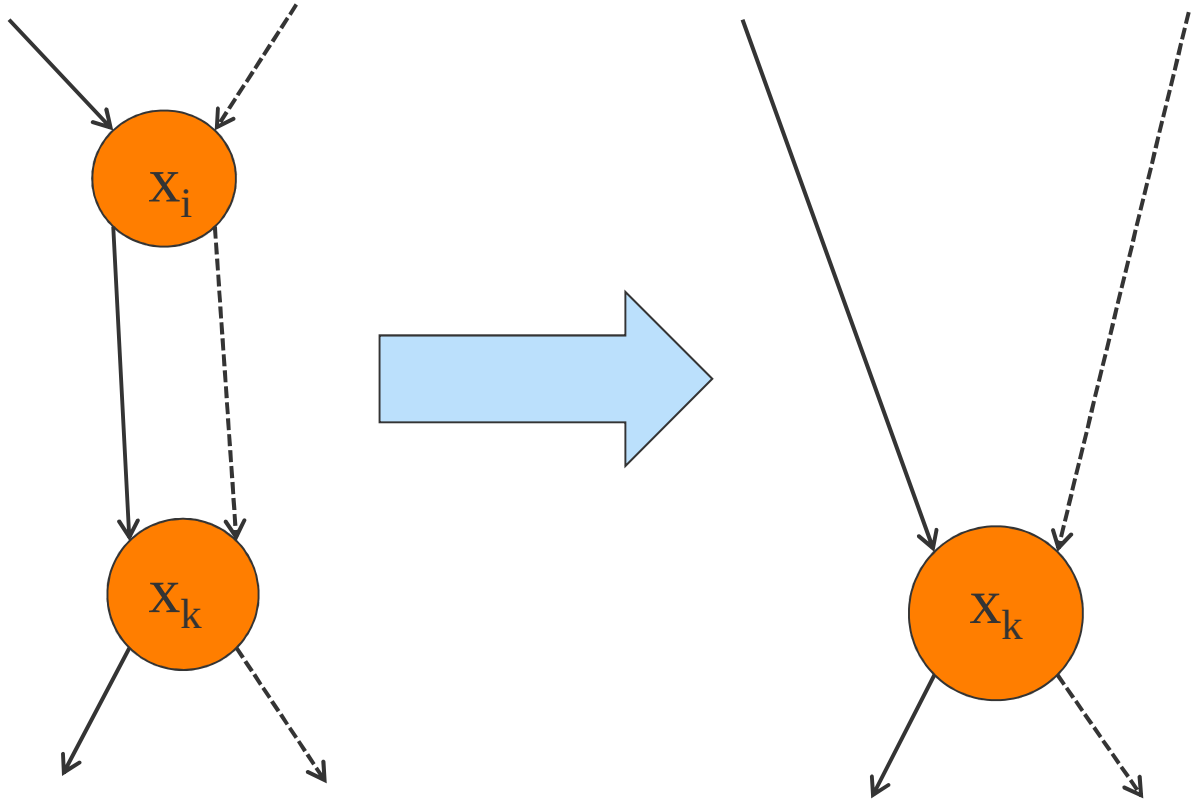
BDD – Reduction Rules

- Umwandlung eines Decision Trees in einen minimalen DAG (Directed Acyclic Graph)
- Anwenden von zwei Regeln
- **Delete Rule**: Löschen unnötiger Zwischenstufen
- **Merge Rule**: Zusammenfassen von identen Teilgraphen
- Ergebnis: Binary Decision Diagram (**BDD**)

  (eigentlich geordnetes BDD, engl. Ordered BDD = OBBD)

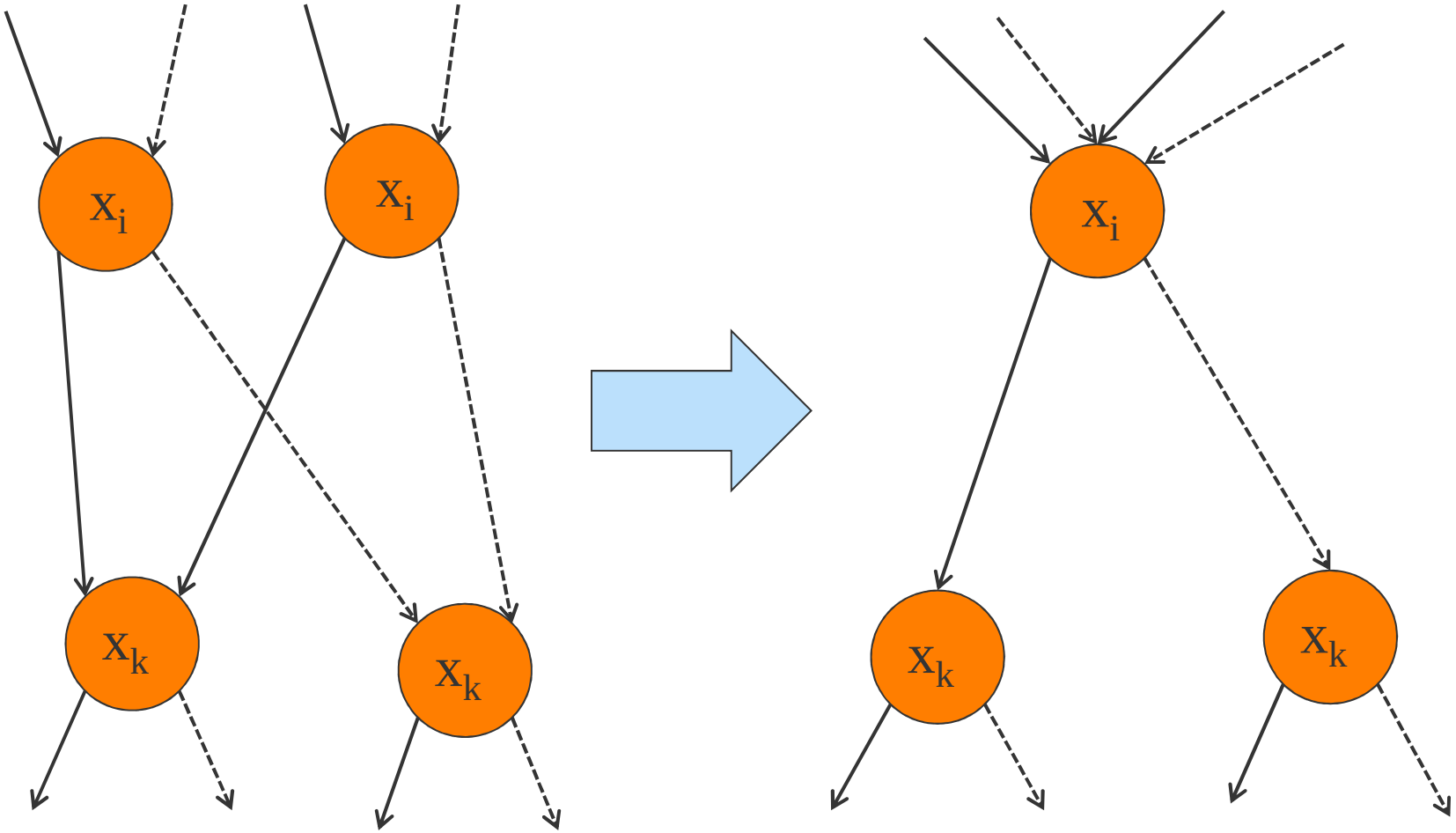
BDD – Reduction Rule

- *Delete Rule*



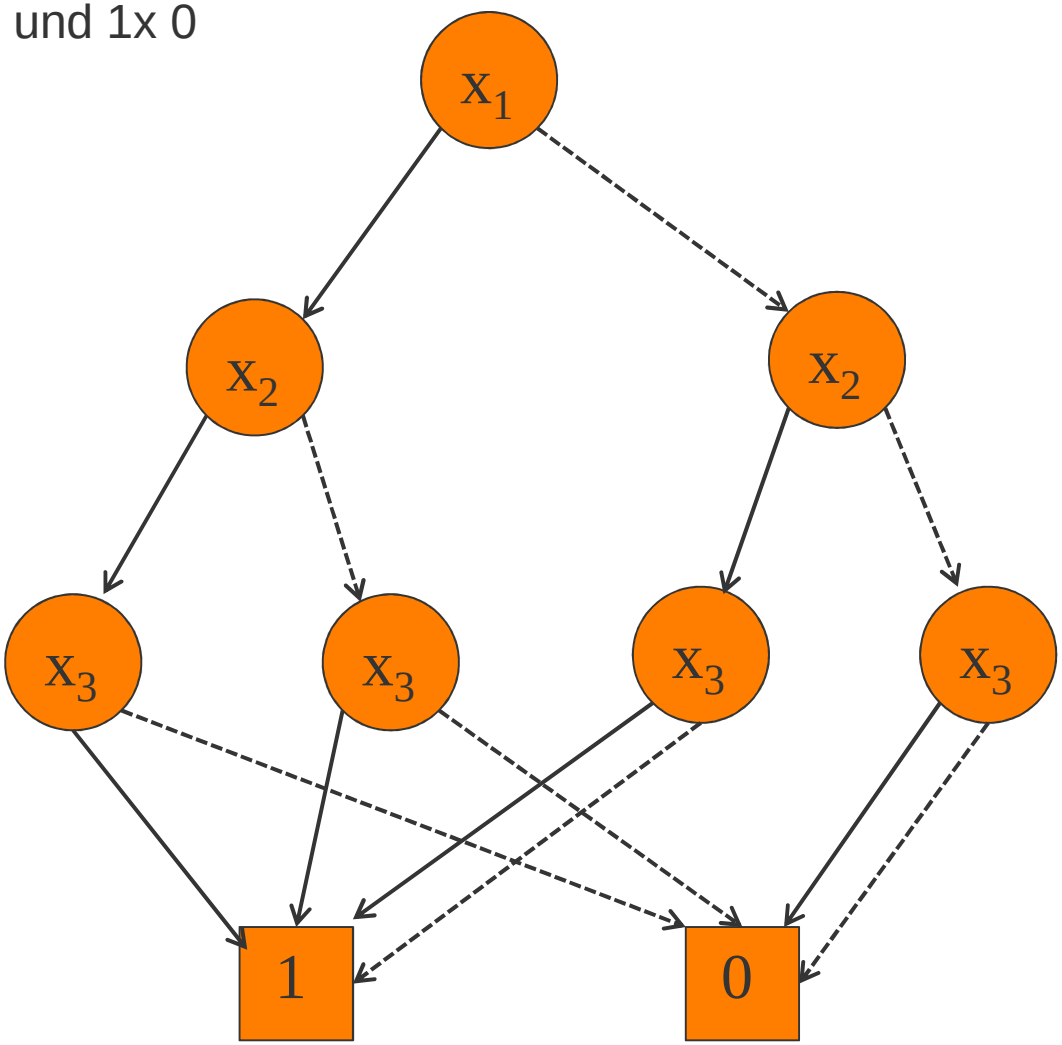
BDD – Reduction Rules

- *Merge Rule*



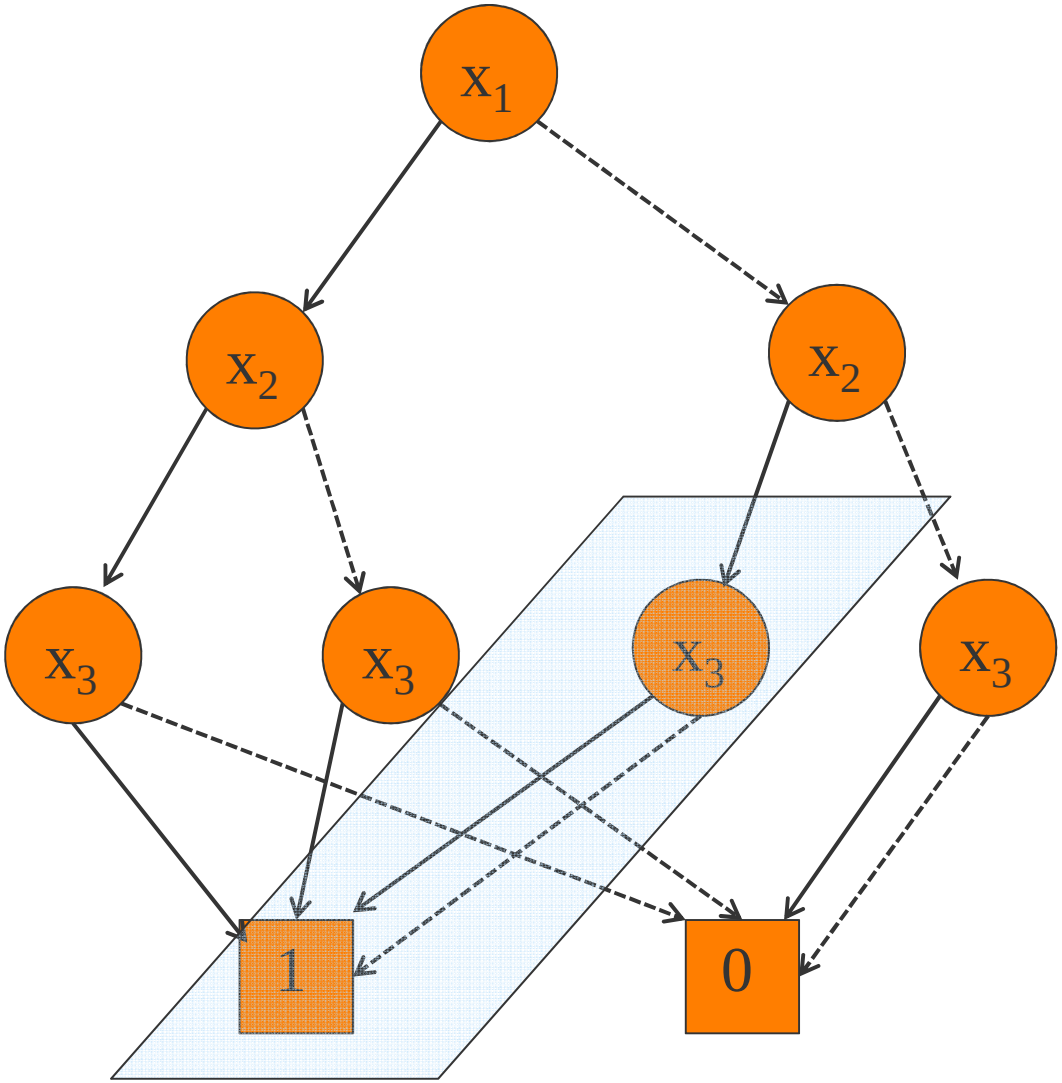
Beispiel

Schritt 1: Nur 1x 1 und 1x 0



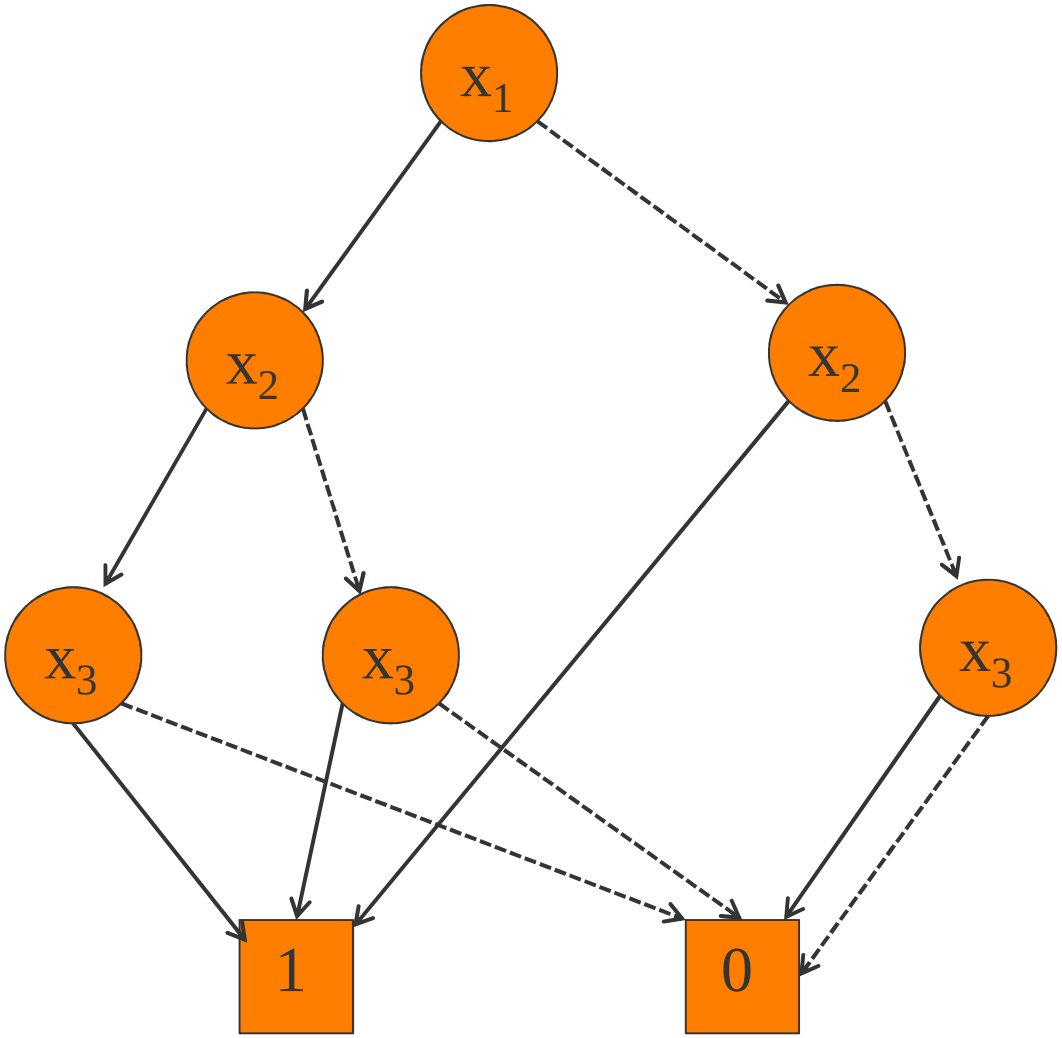
Beispiel

Delete Rule



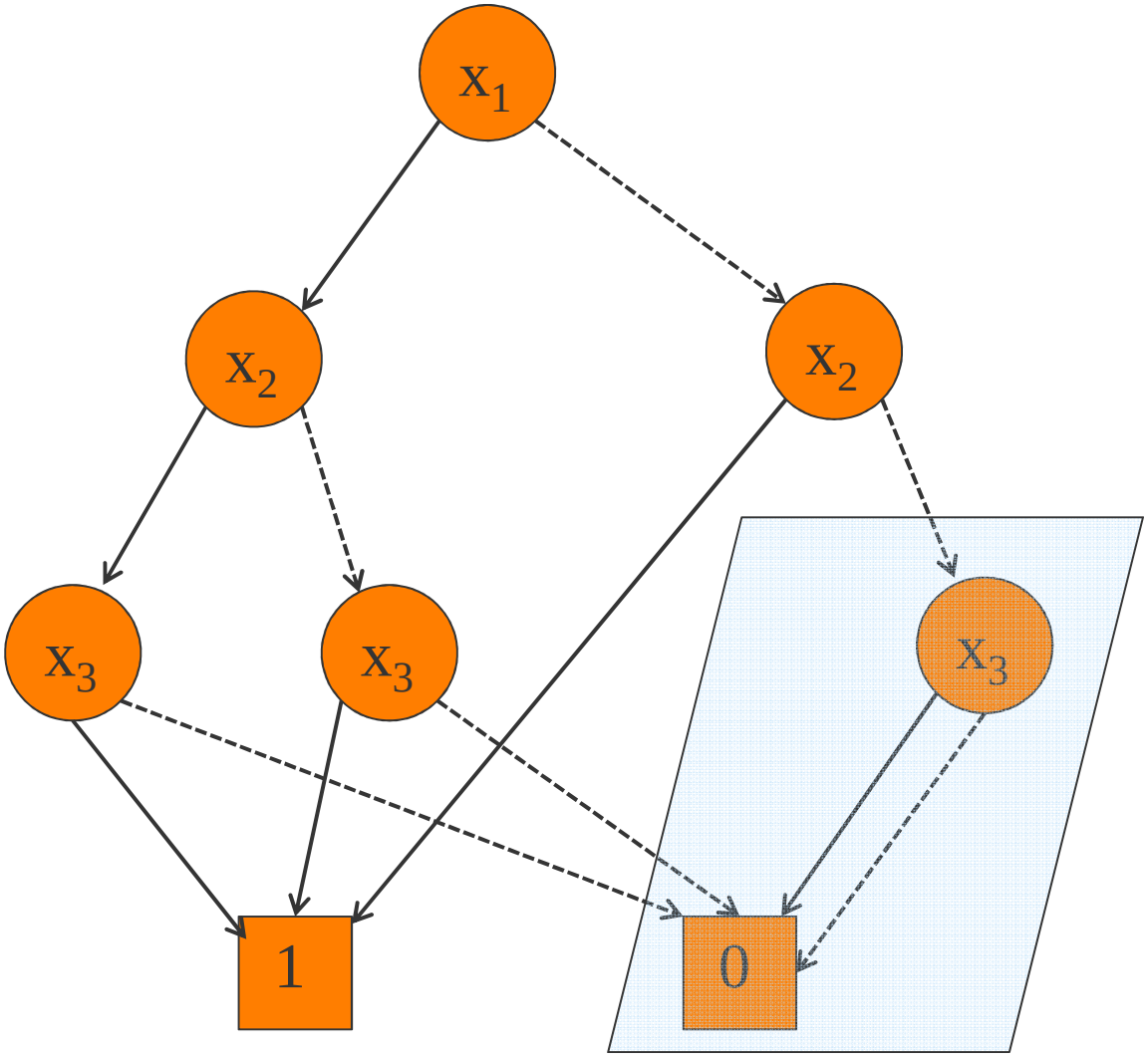
Beispiel

Delete Rule



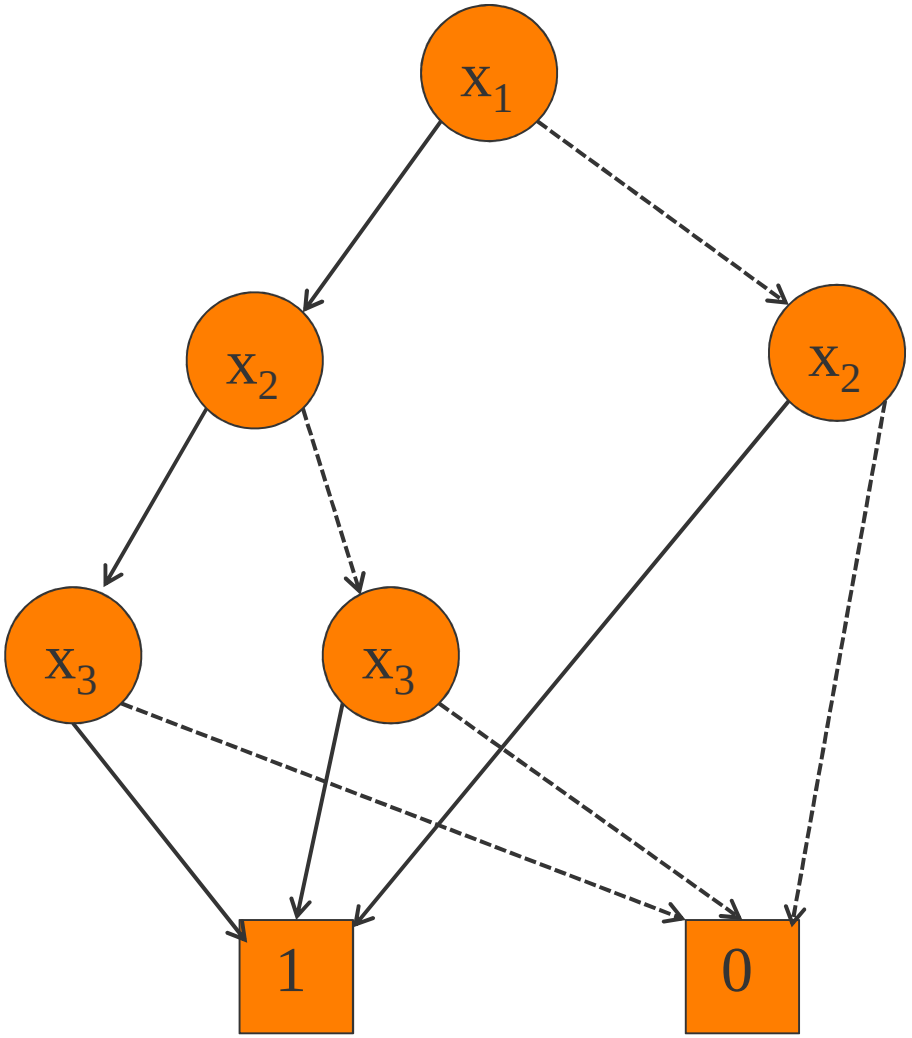
Beispiel

Delete Rule



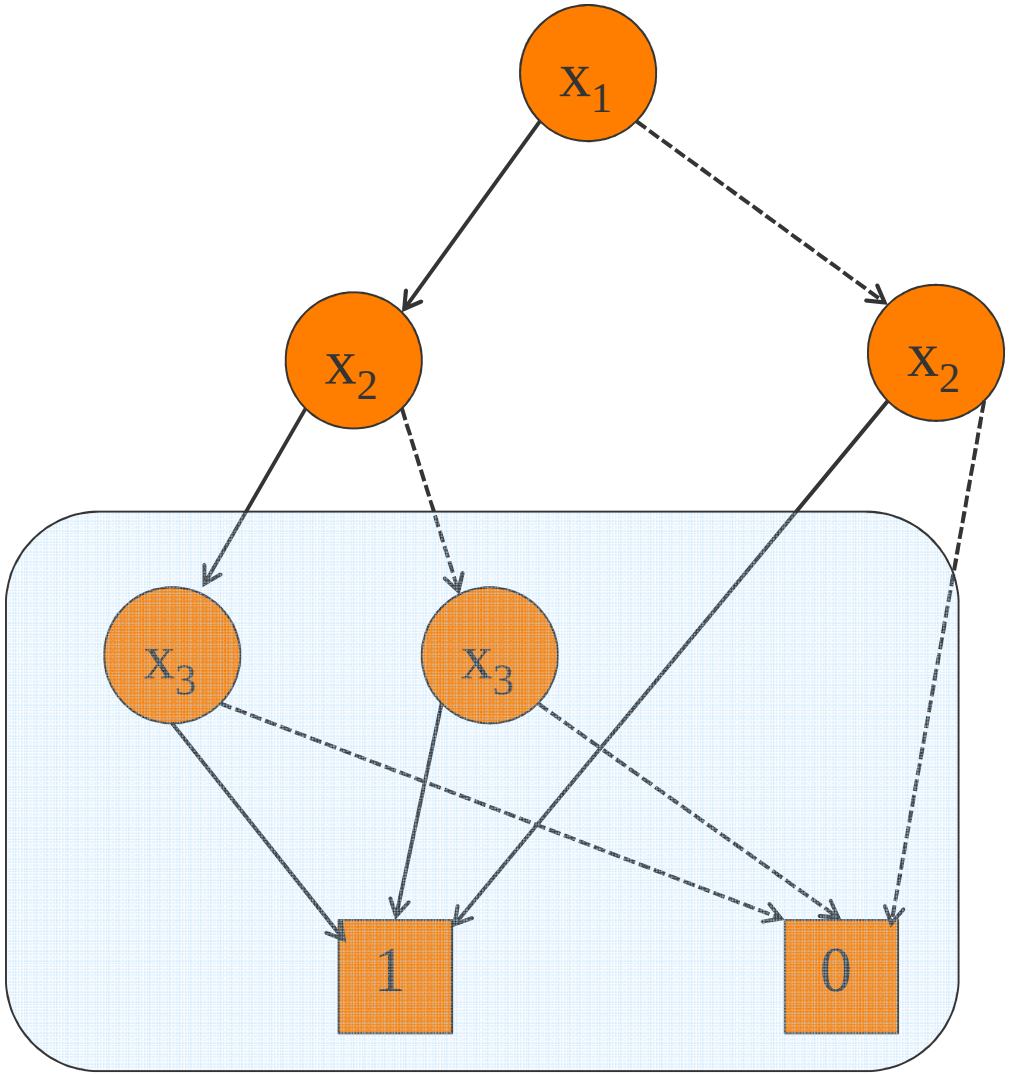
Beispiel

Delete Rule



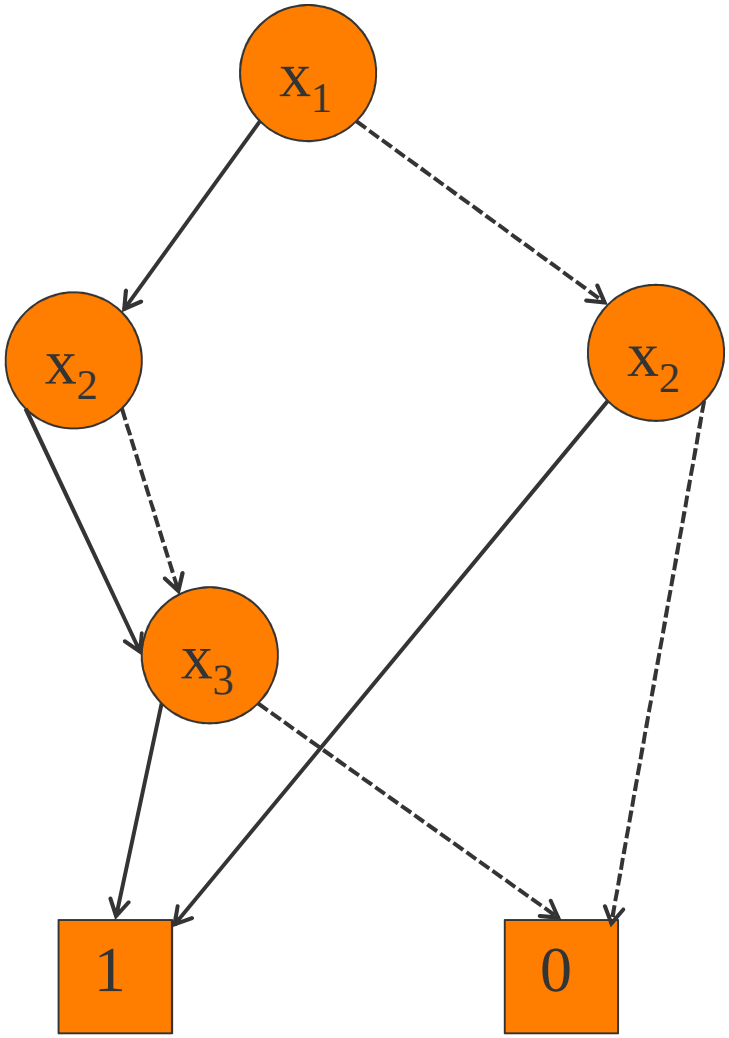
Beispiel

Merge Rule



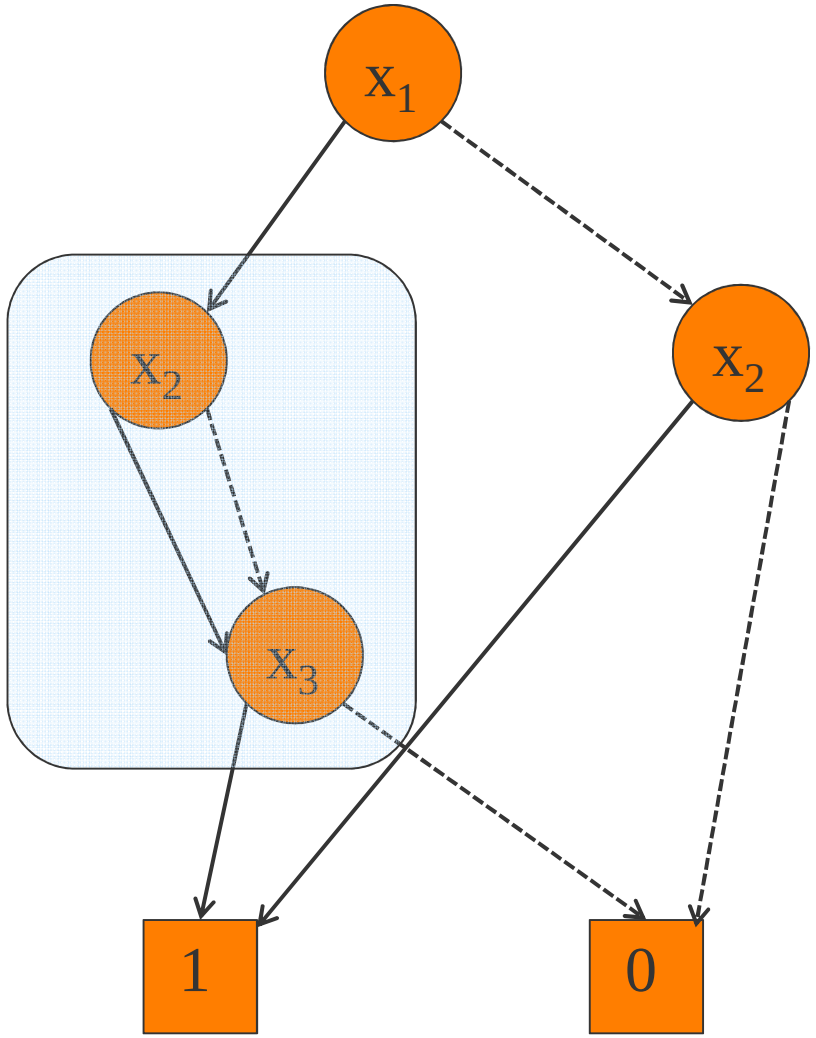
Beispiel

Merge Rule

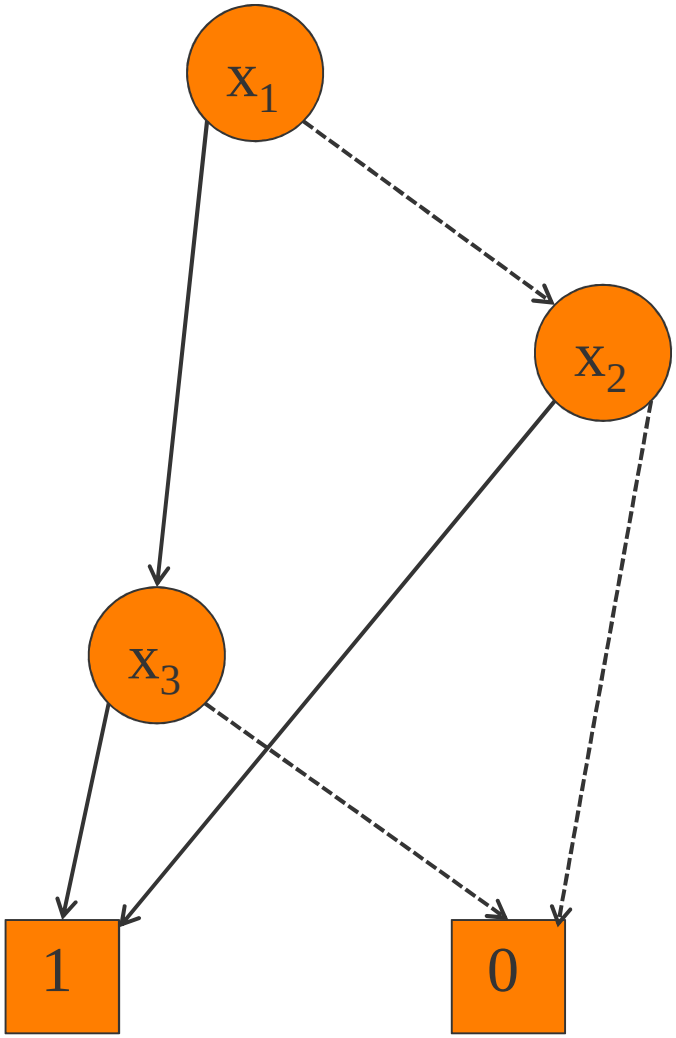


Beispiel

Delete Rule

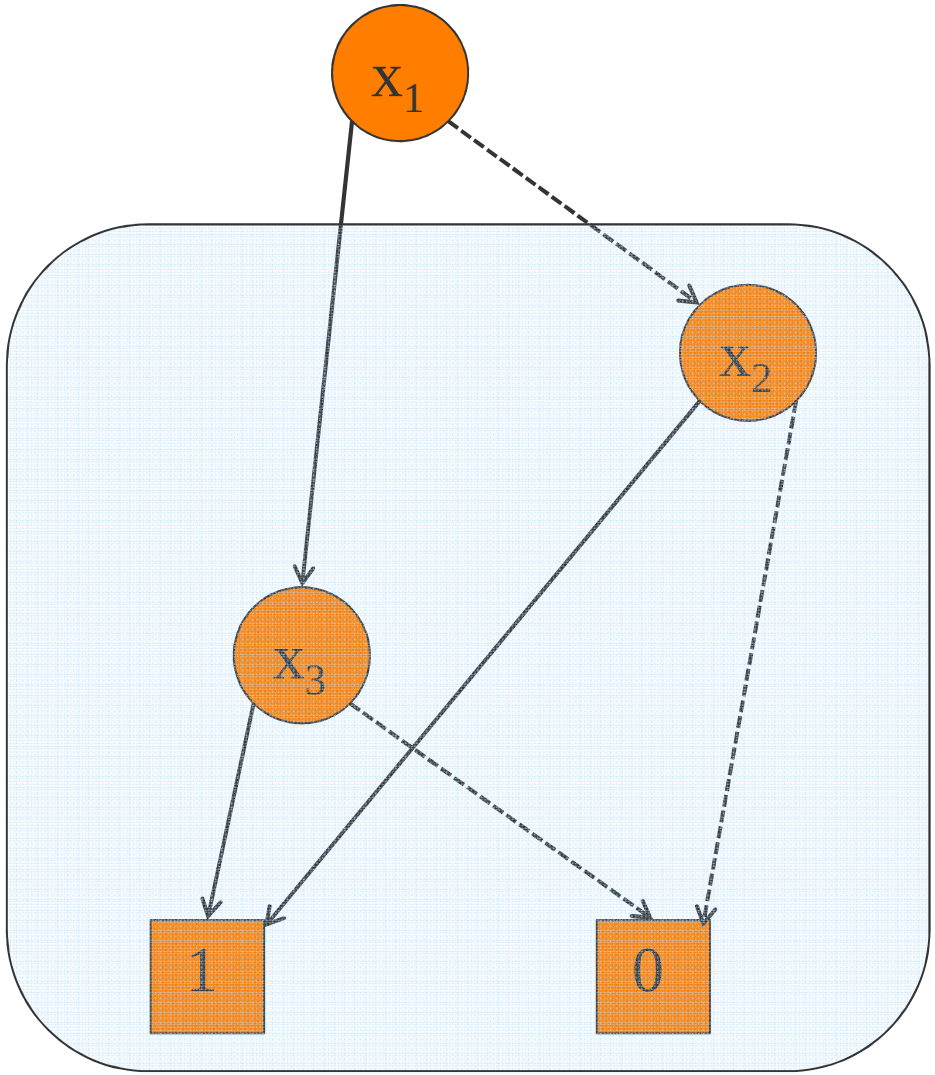


Beispiel



Beispiel

Merge Rule ?

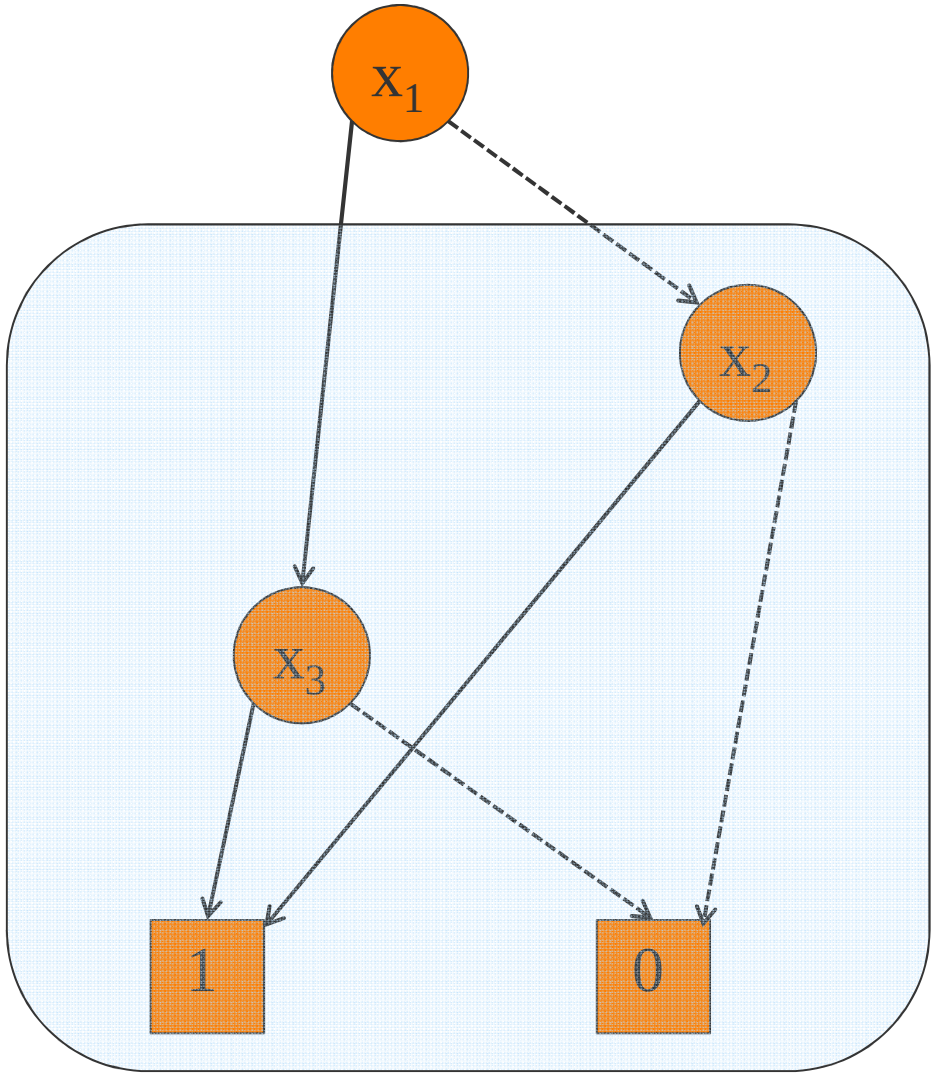


Beispiel

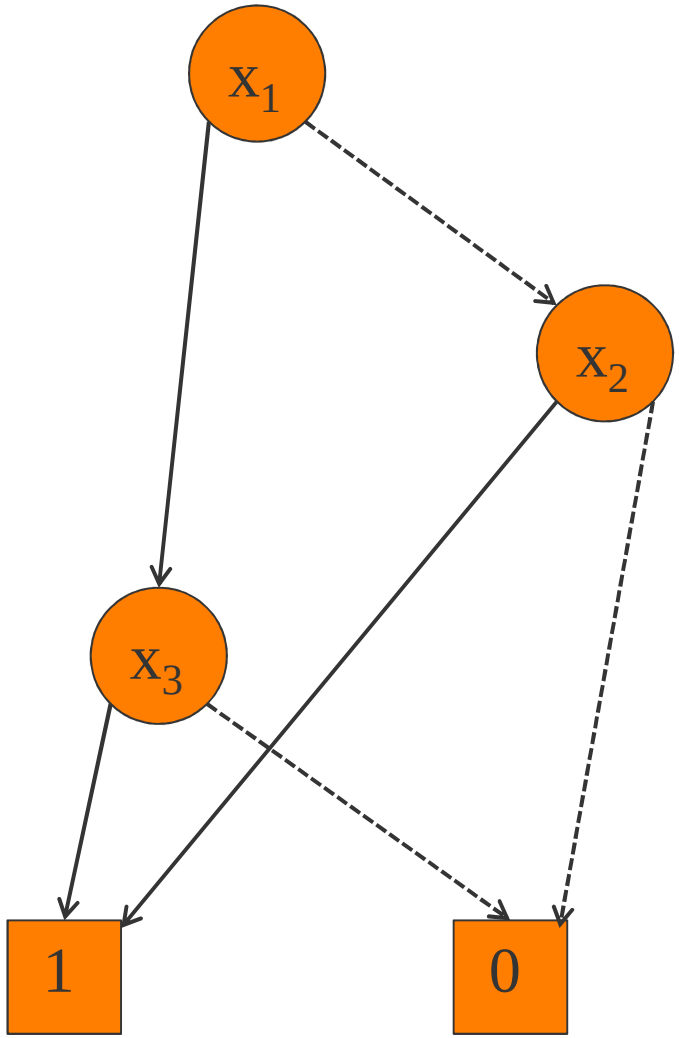
Merge Rule ?

Nein

$x_2 \neq x_3$



Beispiel als ITE



IF x_1 THEN
IF x_3 THEN 1
ELSE 0
ELSE
IF x_2 THEN 1
ELSE 0

Vereinfachte Reduktion (Beads)

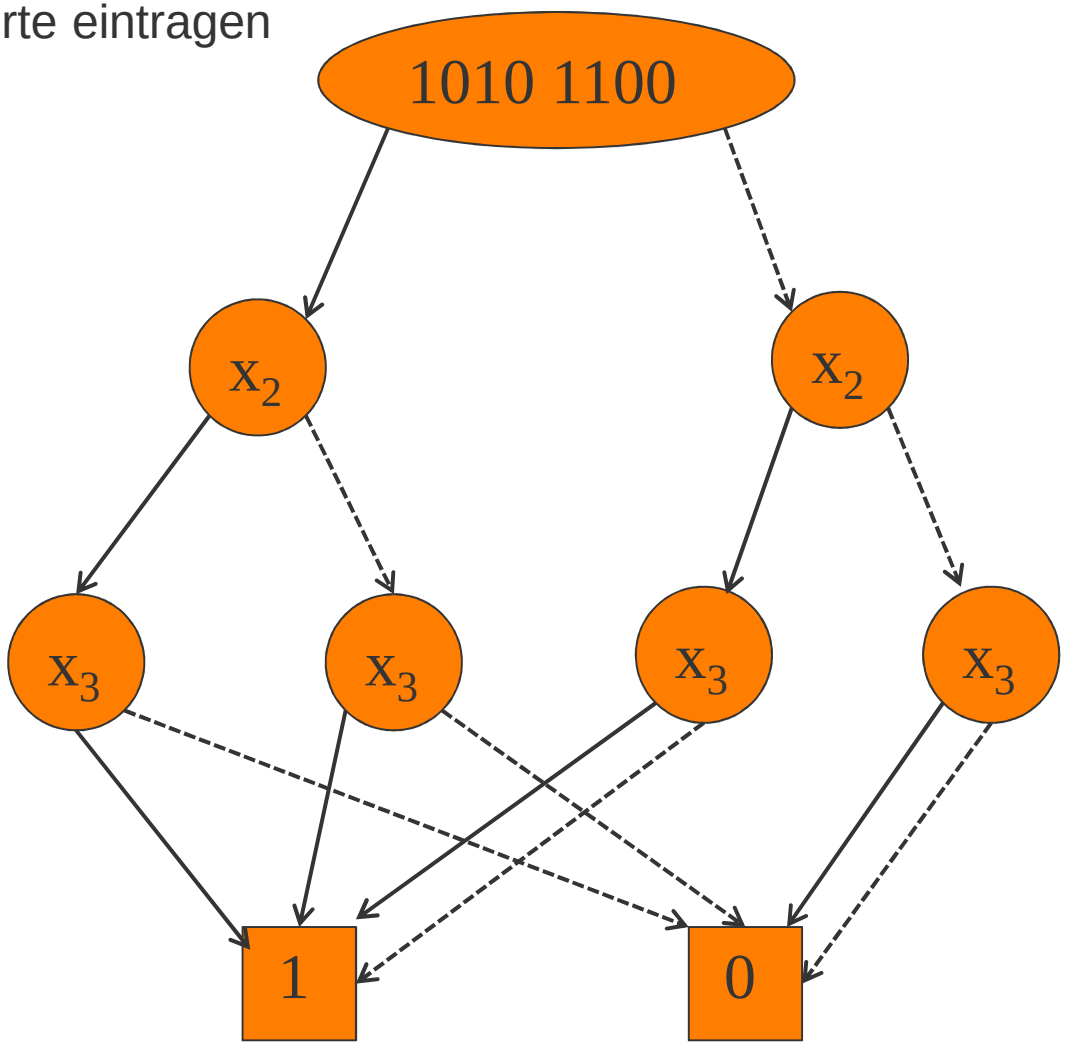
- Bisherige Reduktion im Baum durch Anwendung von
  - Merge-Rule, Delete-Rule
- Vereinfachte Version durch „Aufschreiben“ der Ergebnisse der Wahrheitstabelle als Wort der Länge 2^n wobei n die Anzahl der Eingangsvariablen ist.
  - Z.B.: $f(a,b,c) = 01011100$
- ➔ sogenannte **Beads** (Wörter der Länge 2^n)
- Beads sind binäre Haben nicht die Form **ww**, wobei **w** ein Wort der Länge 2^{n-1} ist.
- Beispiel(n=2): alle Beads der Länge 4 sind

  0001, 0010, 0011, 0100, 0110, 0111,1000, 1001, 1011, 1100, 1101, 1110
- Keine Beads (= Squares) der Länge 4 sind

  0000, 0101, 1010, 1111

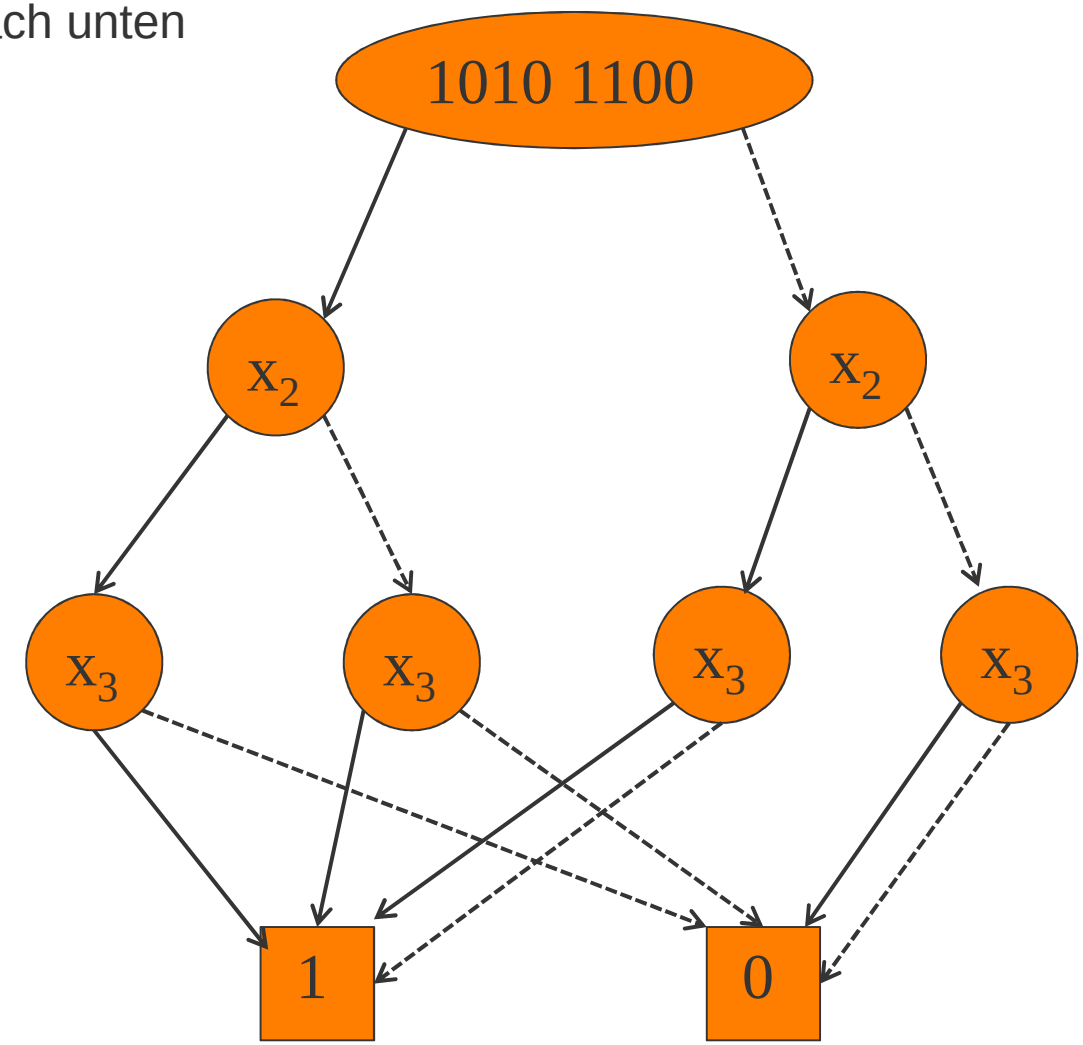
Beispiel via Beads

In x_1 Wahrheitswerte eintragen



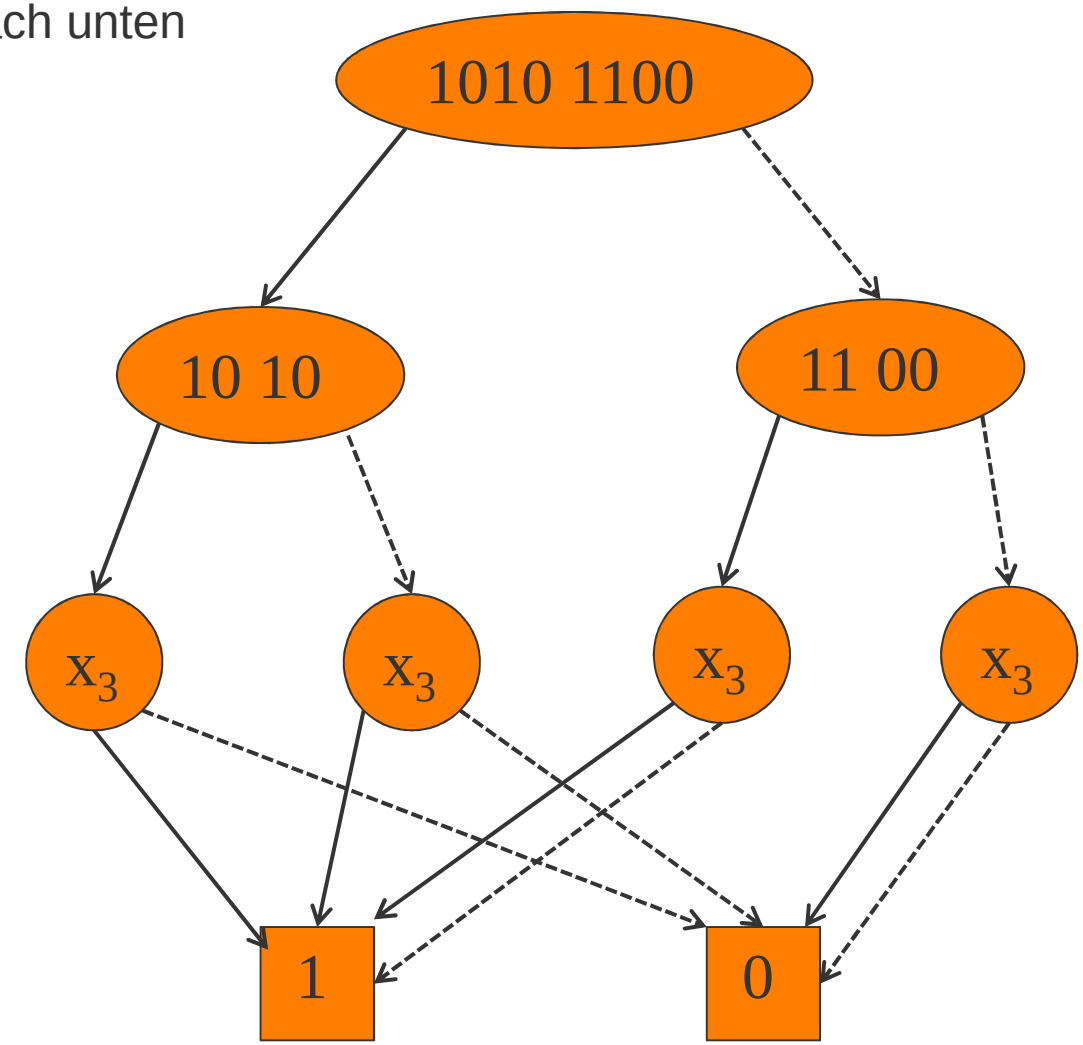
Beispiel via Beads

Halbieren und nach unten übertragen



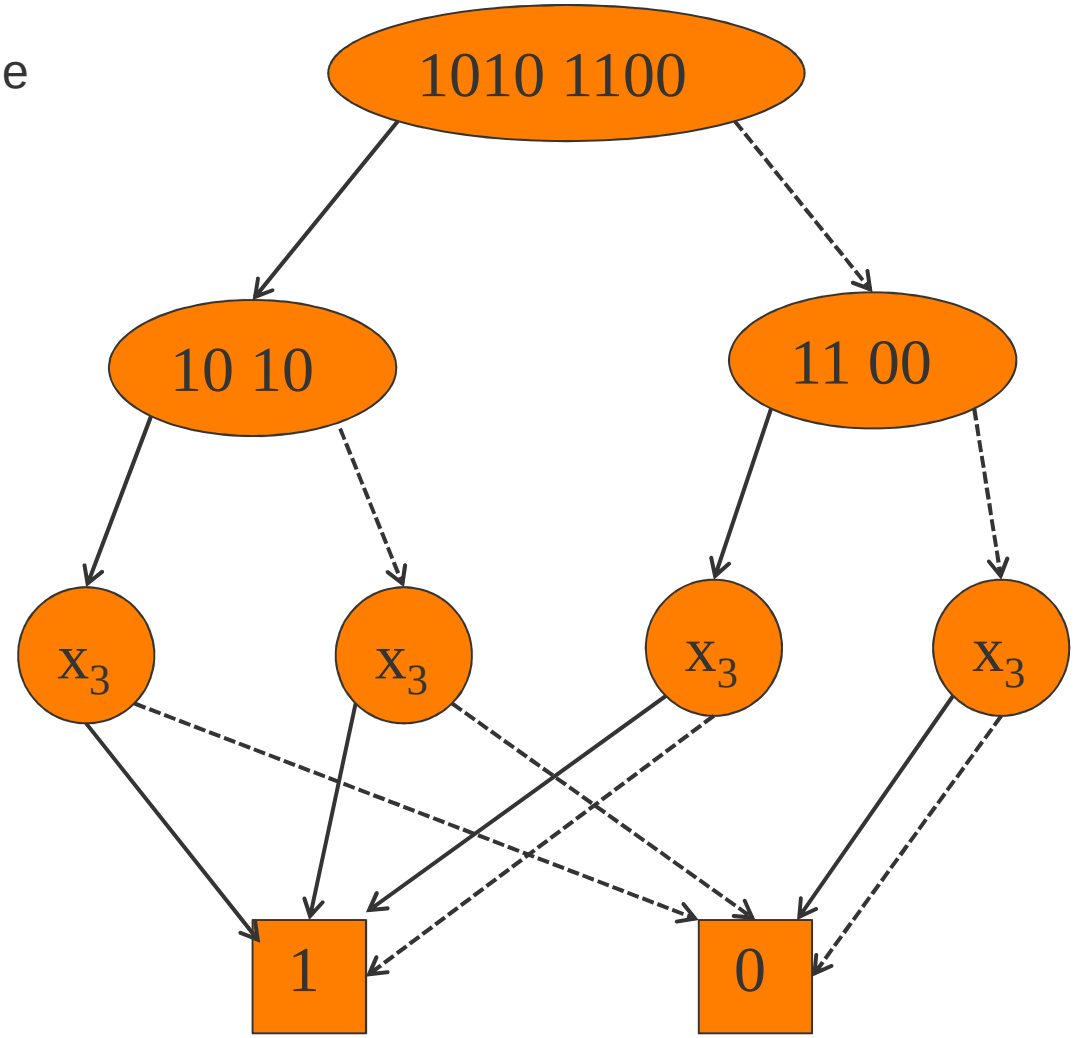
Beispiel via Beads

Halbieren und nach unten übertragen



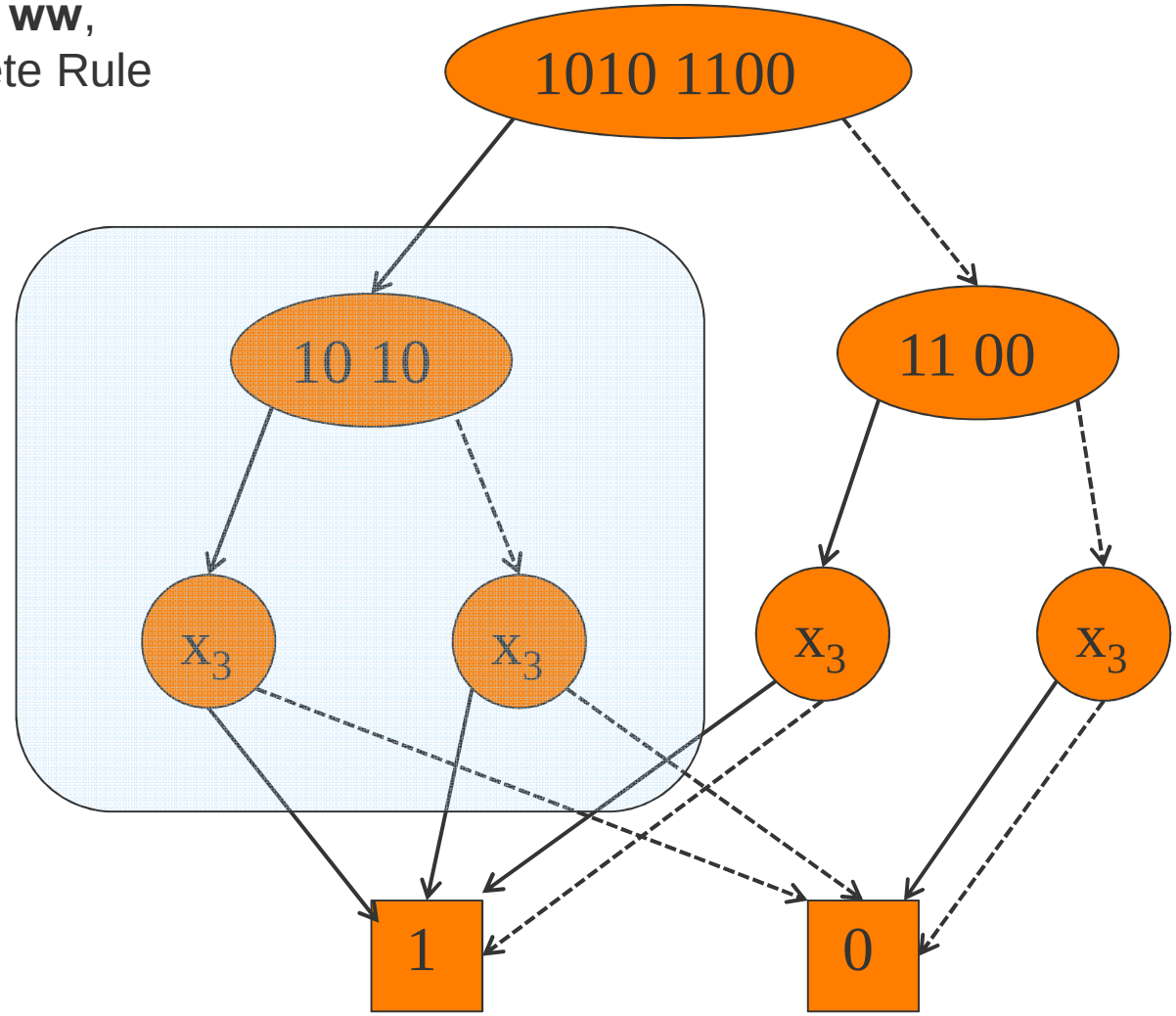
Beispiel via Beads

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



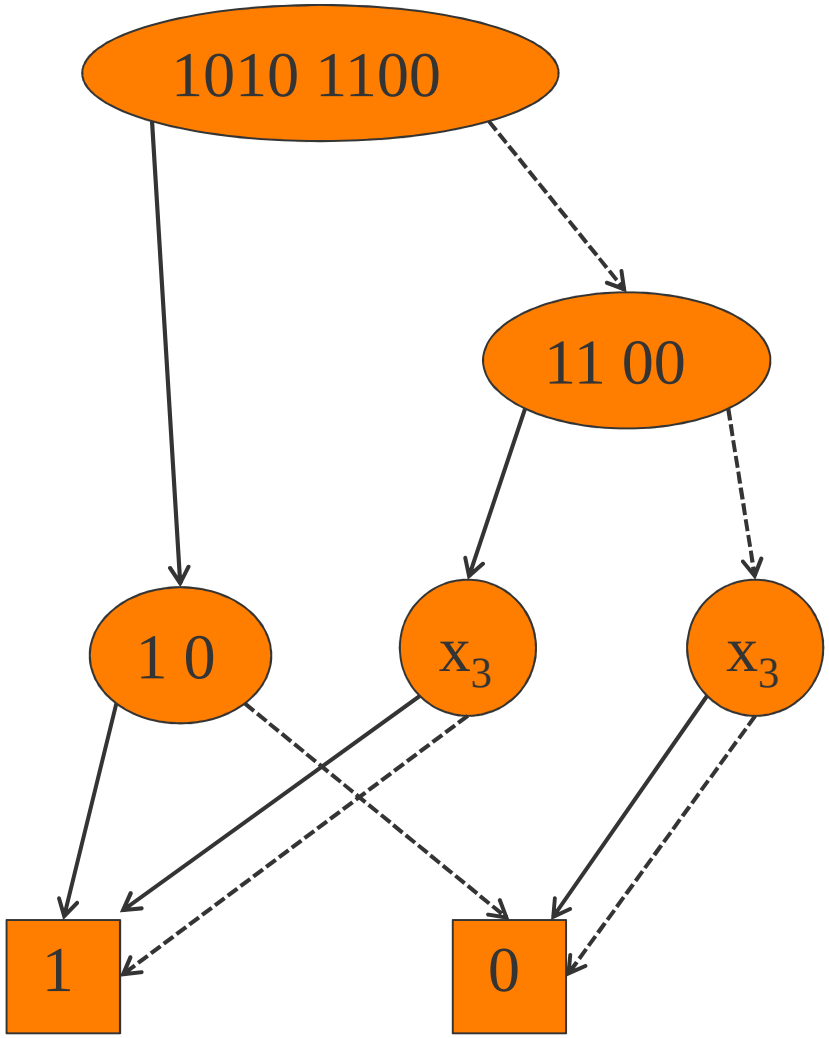
Beispiel via Beads

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



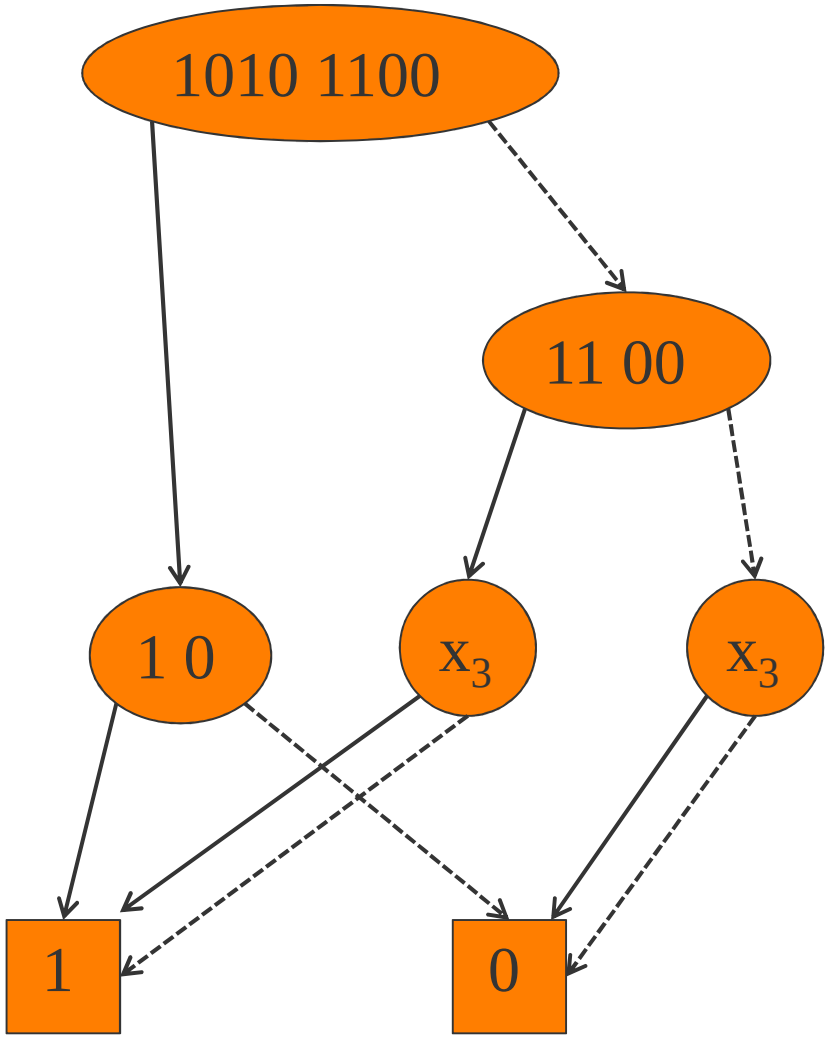
Beispiel via Beads

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



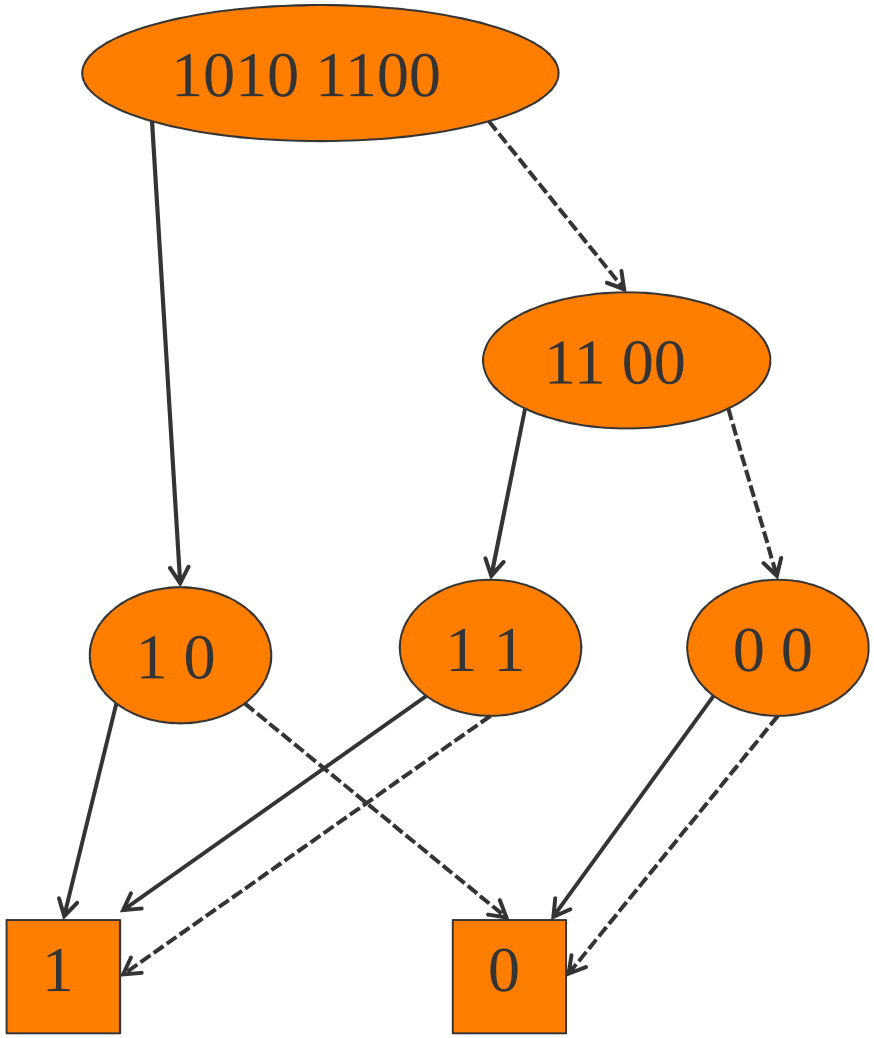
Beispiel via Beads

Halbieren und nach unten



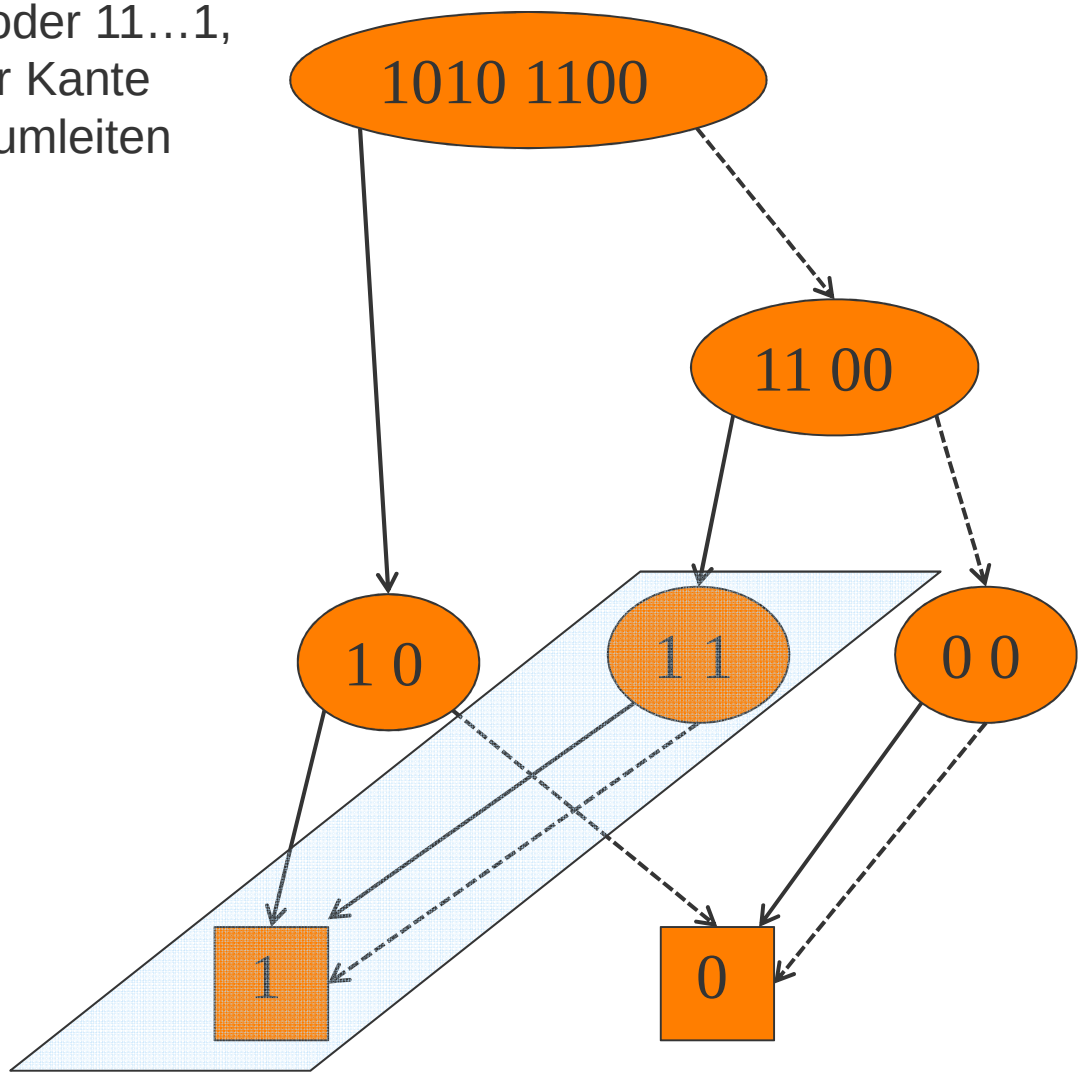
Beispiel via Beads

Halbieren und nach unten



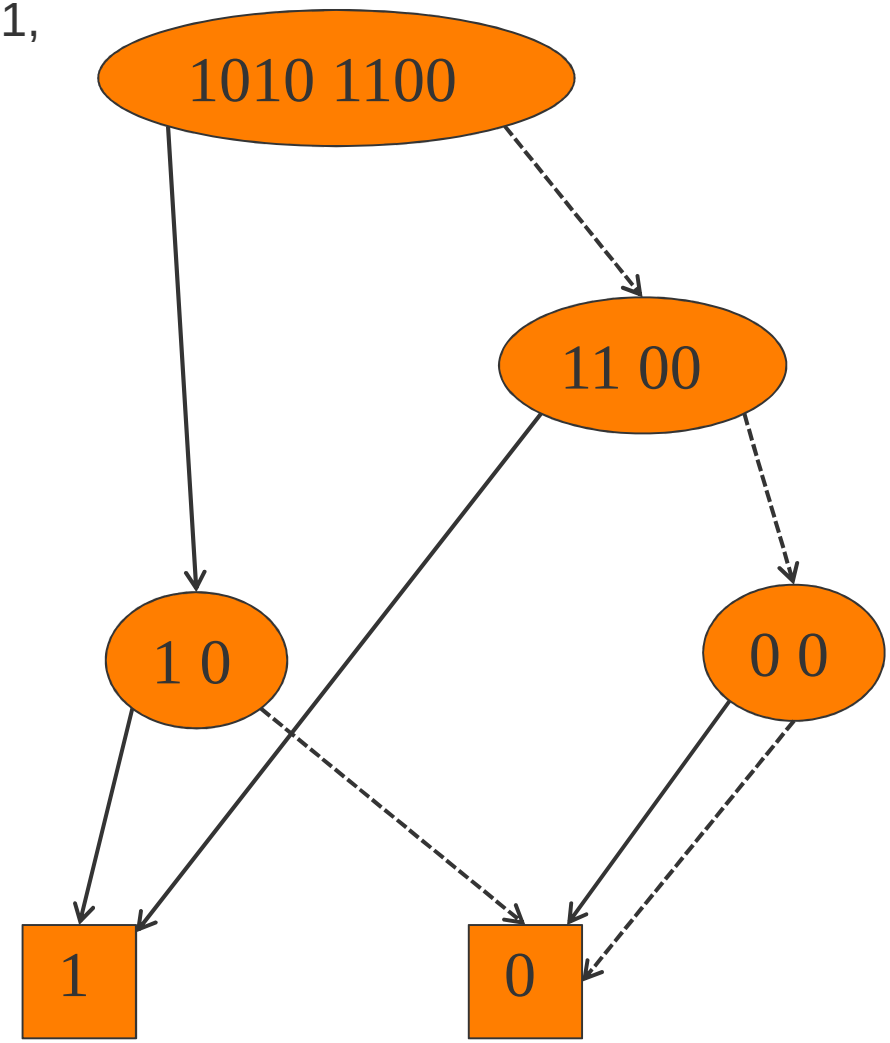
Beispiel via Beads

Falls gleich 00…0 oder 11…1,
Je nach eigehender Kante
direkt auf 0 oder 1 umleiten



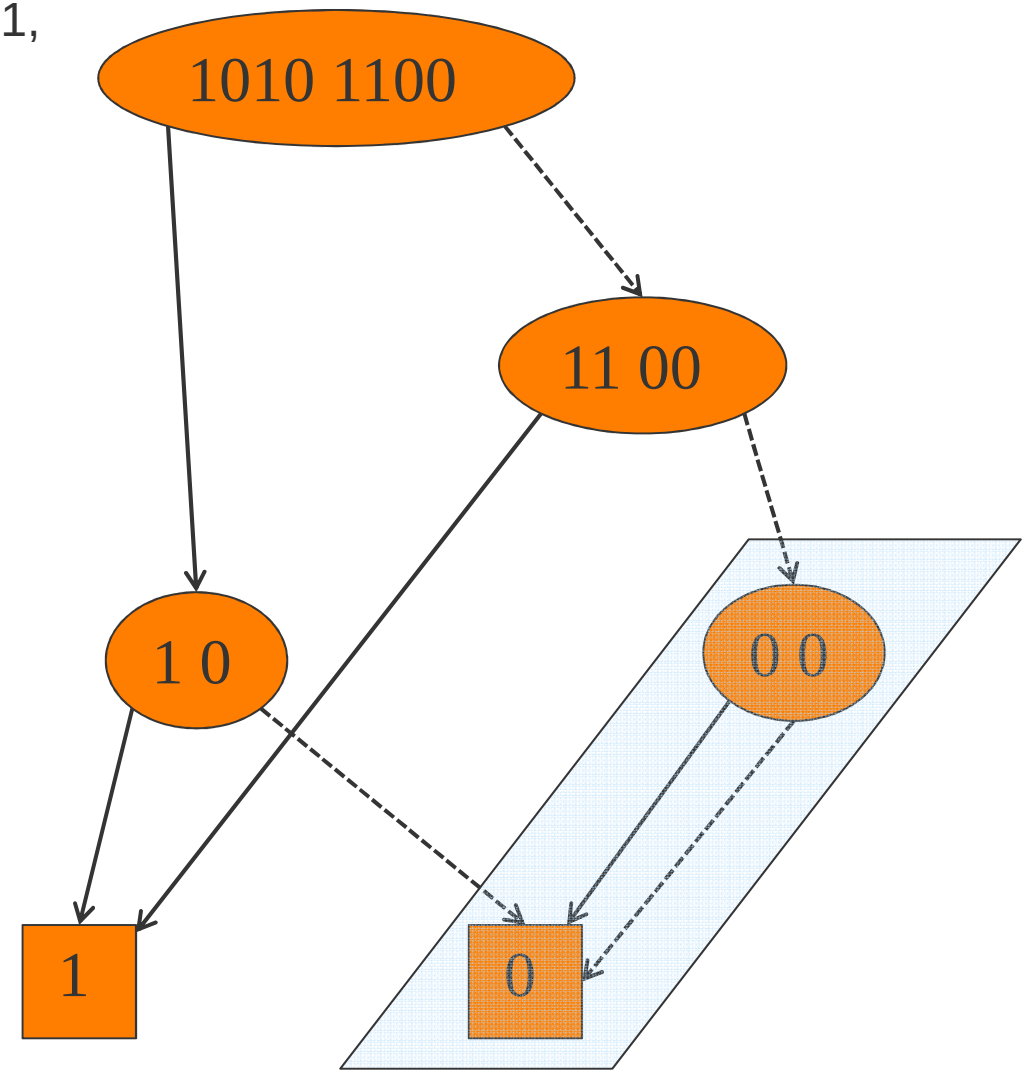
Beispiel via Beads

Falls gleich 00…0 oder 11…1,
Je nach eigehender Kante
direkt auf 0 oder 1 umleiten



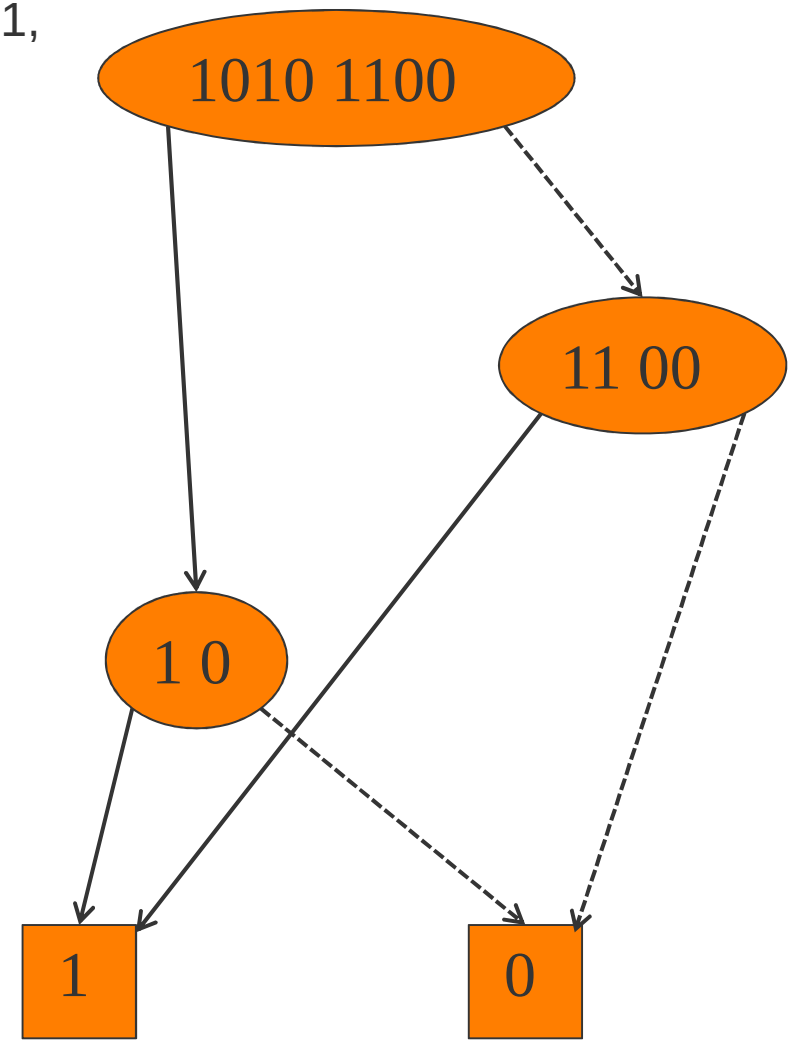
Beispiel via Beads

Falls gleich 00…0 oder 11…1,
Je nach eigehender Kante
direkt auf 0 oder 1 umleiten



Beispiel via Beads

Falls gleich 00…0 oder 11…1,
Je nach eigehender Kante
direkt auf 0 oder 1 umleiten

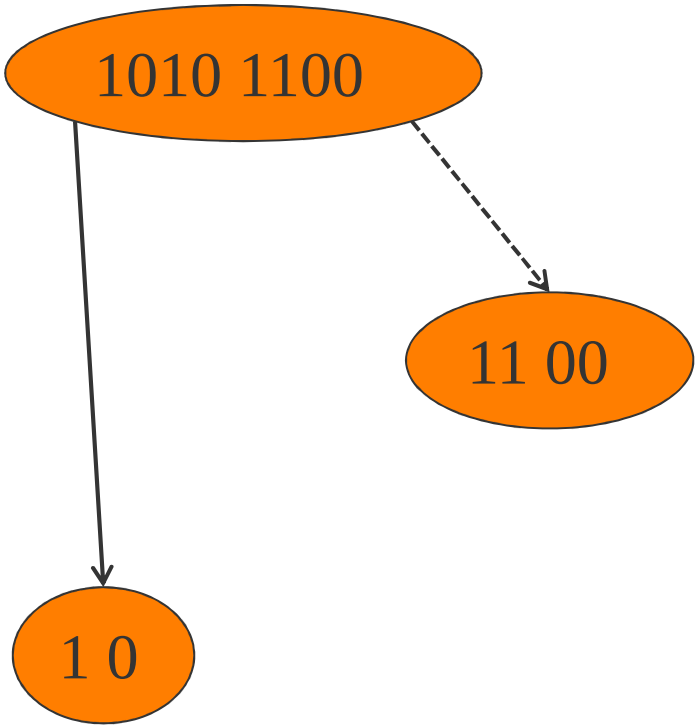


Beispiel via Beads ohne Decision Tree

- Wahrheitstabelle eintragen

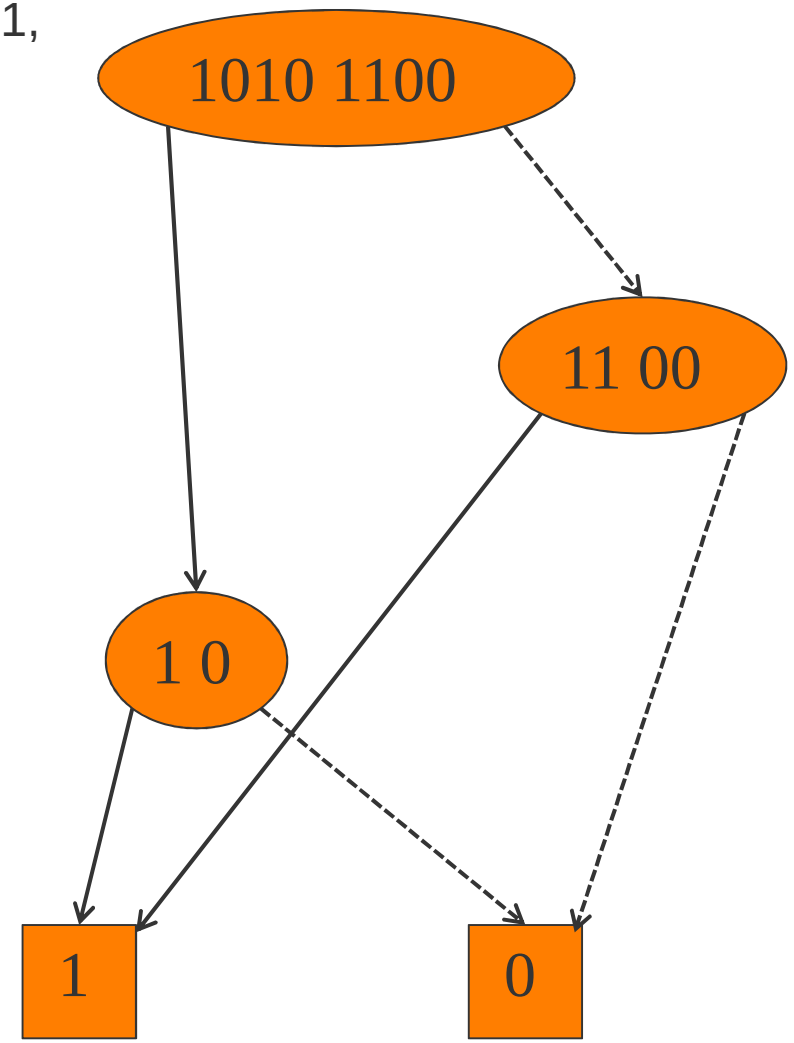
Beispiel via Beads ohne Decision Tree

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



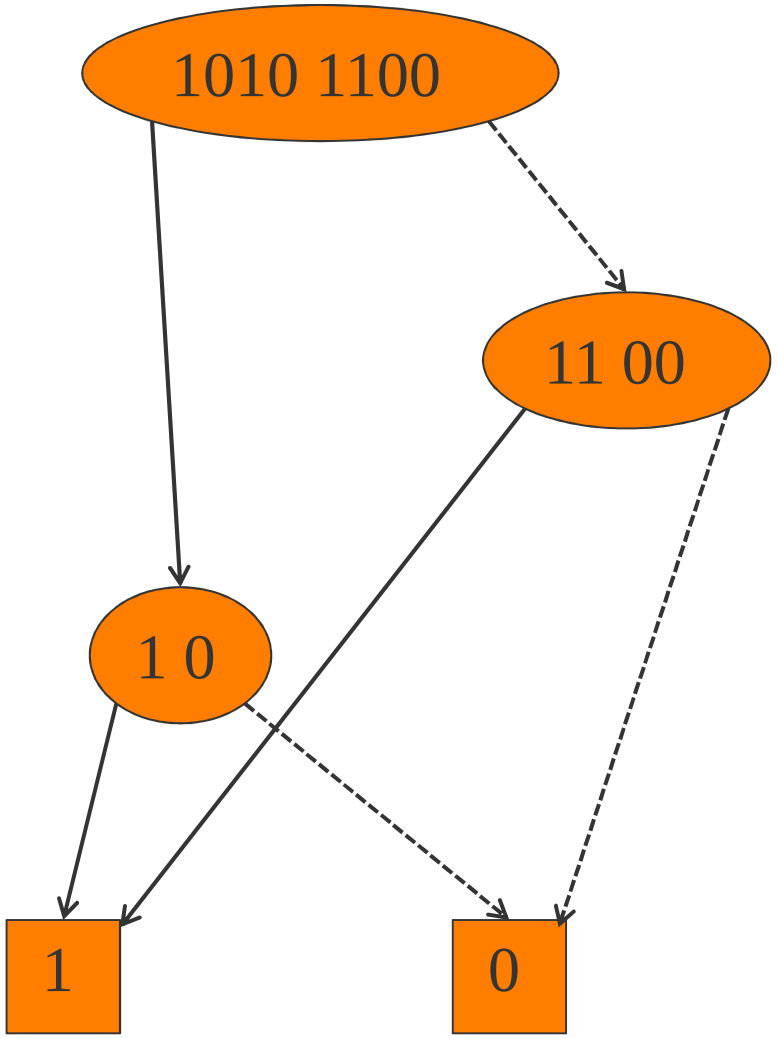
Beispiel via Beads ohne Decision Tree

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



Beispiel via Beads ohne Decision Tree

Reihenfolge der Variablen ergibt sich aus der Wahrheitstabelle



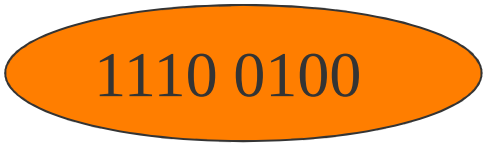
Variablenreihenfolge

- Wählen andere Reihenfolge

x_3	x_2	x_1	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

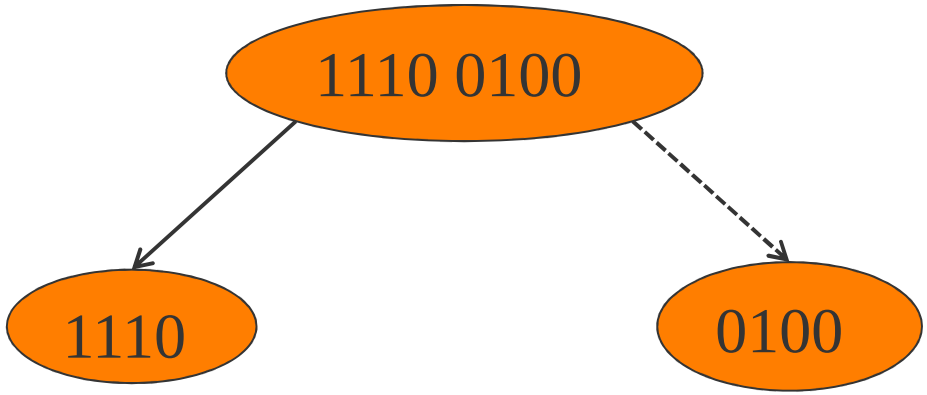
Beispiel

- Wahrheitstabelle eintragen



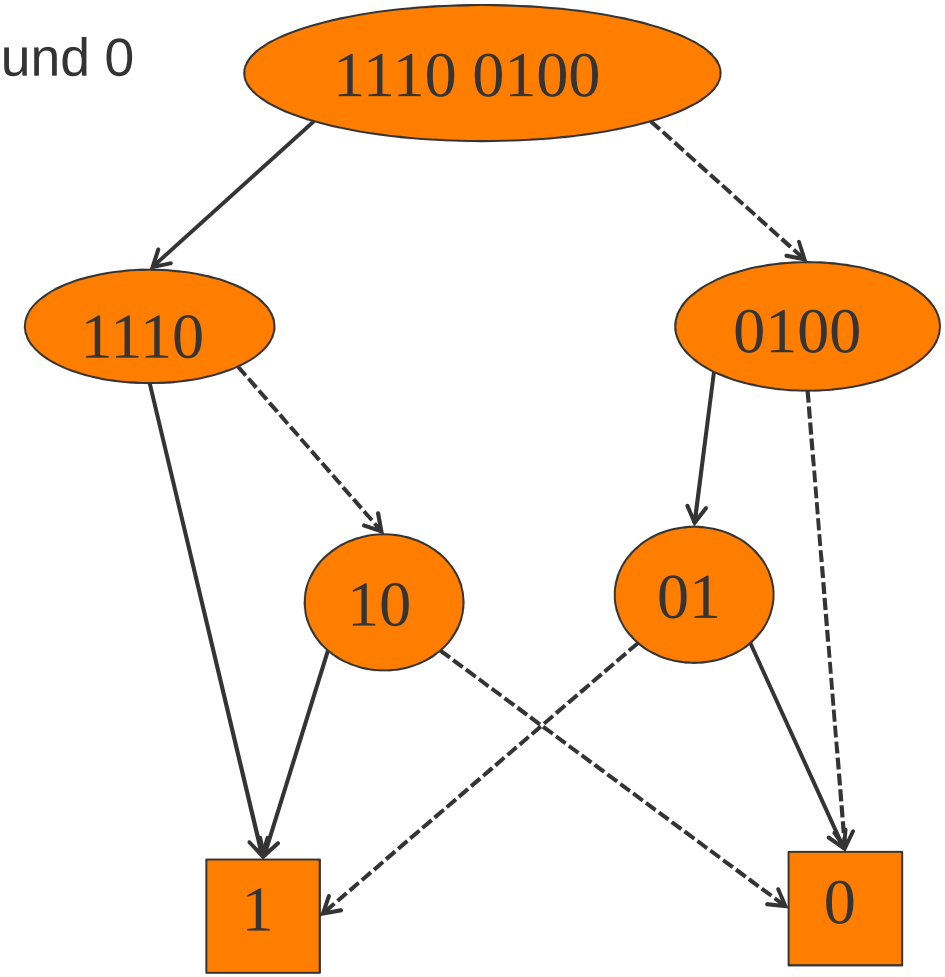
Beispiel

- Halbieren und nach unten



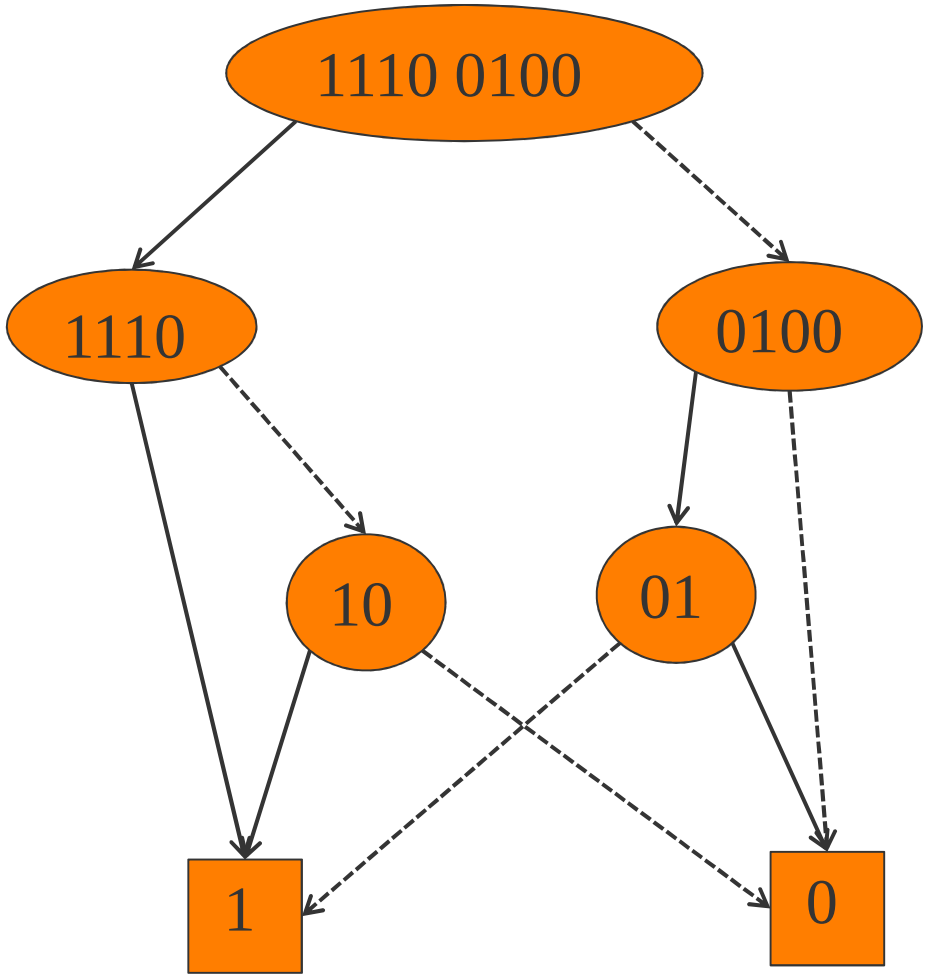
Beispiel

- Halbieren und nach unten
- 11 und 00 auf 1 und 0 umleiten



Beispiel

- Benötigt mehr Knoten!



Variablenreihenfolge

- Für **n** Variablen gibt es **n!** unterschiedliche Permutationen (Permutationen)
- Z.B. $\pi = (a, b, c, d)$, $\pi' = (c, a, b, d)$, ...
- $\Longrightarrow$ Minimum-Finden ist schwer!

Größe von BDDs

- Es gibt Boolesche Funktionen, sodass unabhängig von der Variablenreihenfolge alle BDDs exponentielle Größe haben.
- Die meisten praktisch relevanten Funktionen haben geringe Größe.
- Bekannte schwere Funktion: *Multiplikation.*
- Exponentiell große disjunktive Form aber polynomielle BDD-Größe: *Majority-Funktion.*

Eigenschaften von BDDs

- In einem BDD kommen **alle** Beads der Wahrheitstabelle

  und **nur** diese vor.

- Darstellung einer bestimmten Booleschen Funktion ist bei gegebener

  Variablenreihenfolge **isomorph (eindeutig).**

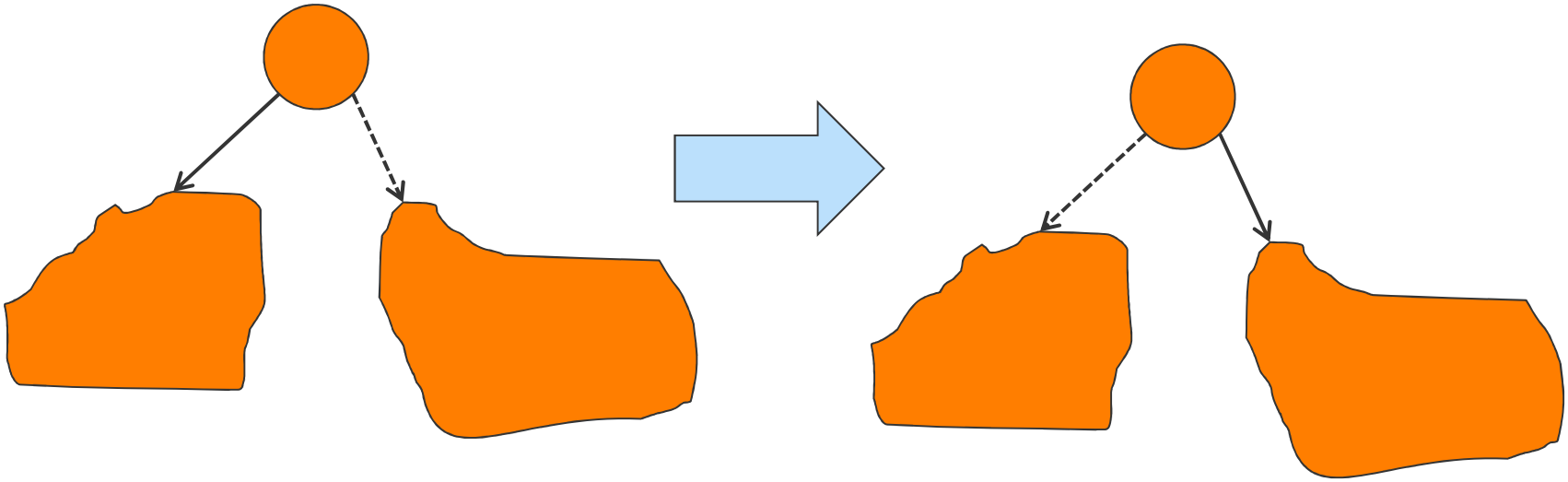
- Zwei isomorphe OBDDs beschreiben äquivalente Boolesche Ausdrücke

- Typische Funktionen sind einfach mittels BDD-Darstellung zu

  realisieren.

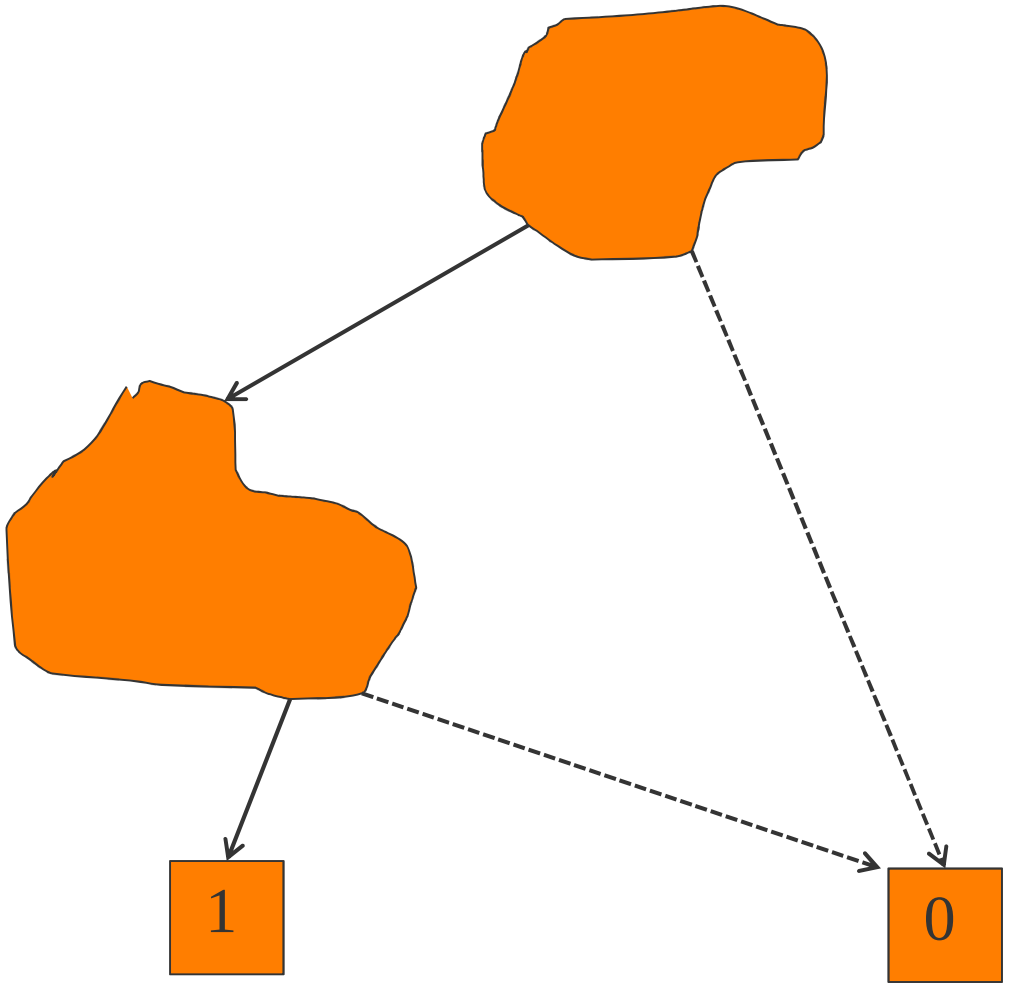
Typische Funktionen auf BDDs

- **Negation**: Kantentypen vertauschen



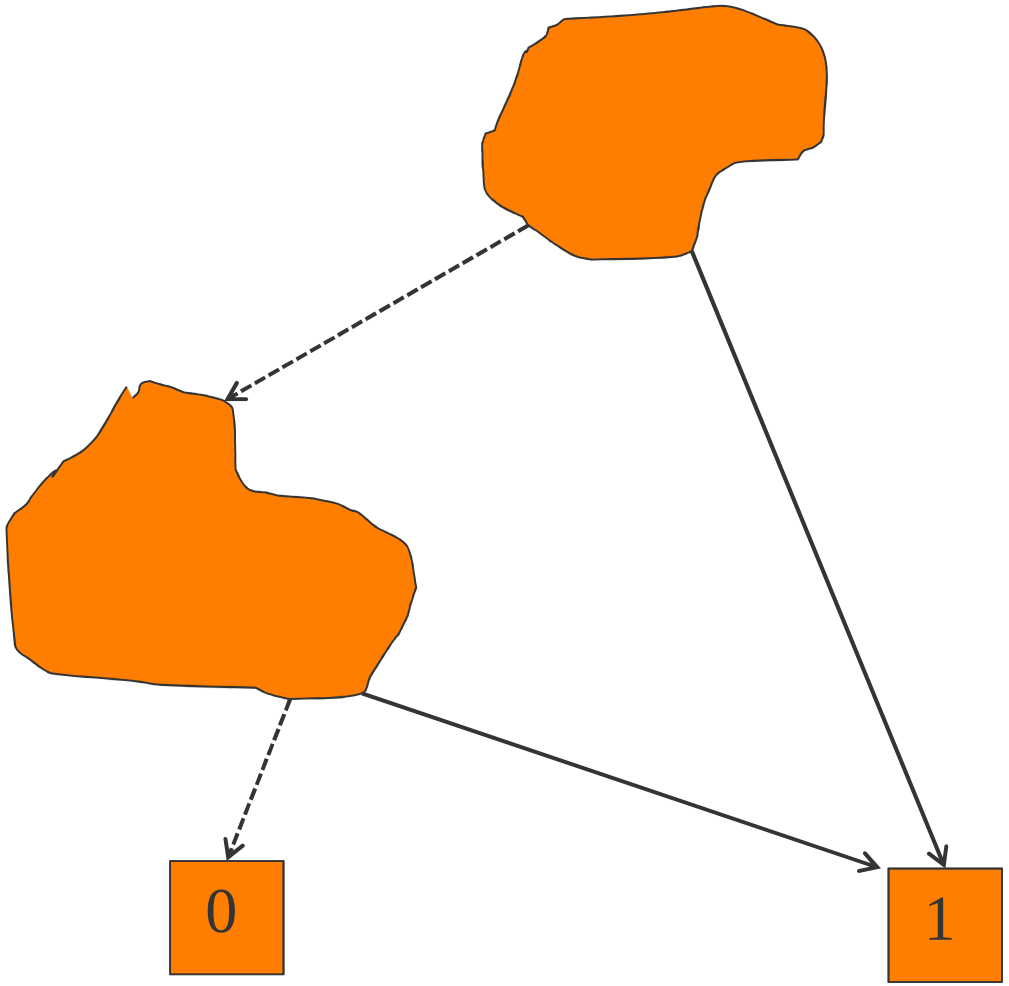
Typische Funktionen

- UND



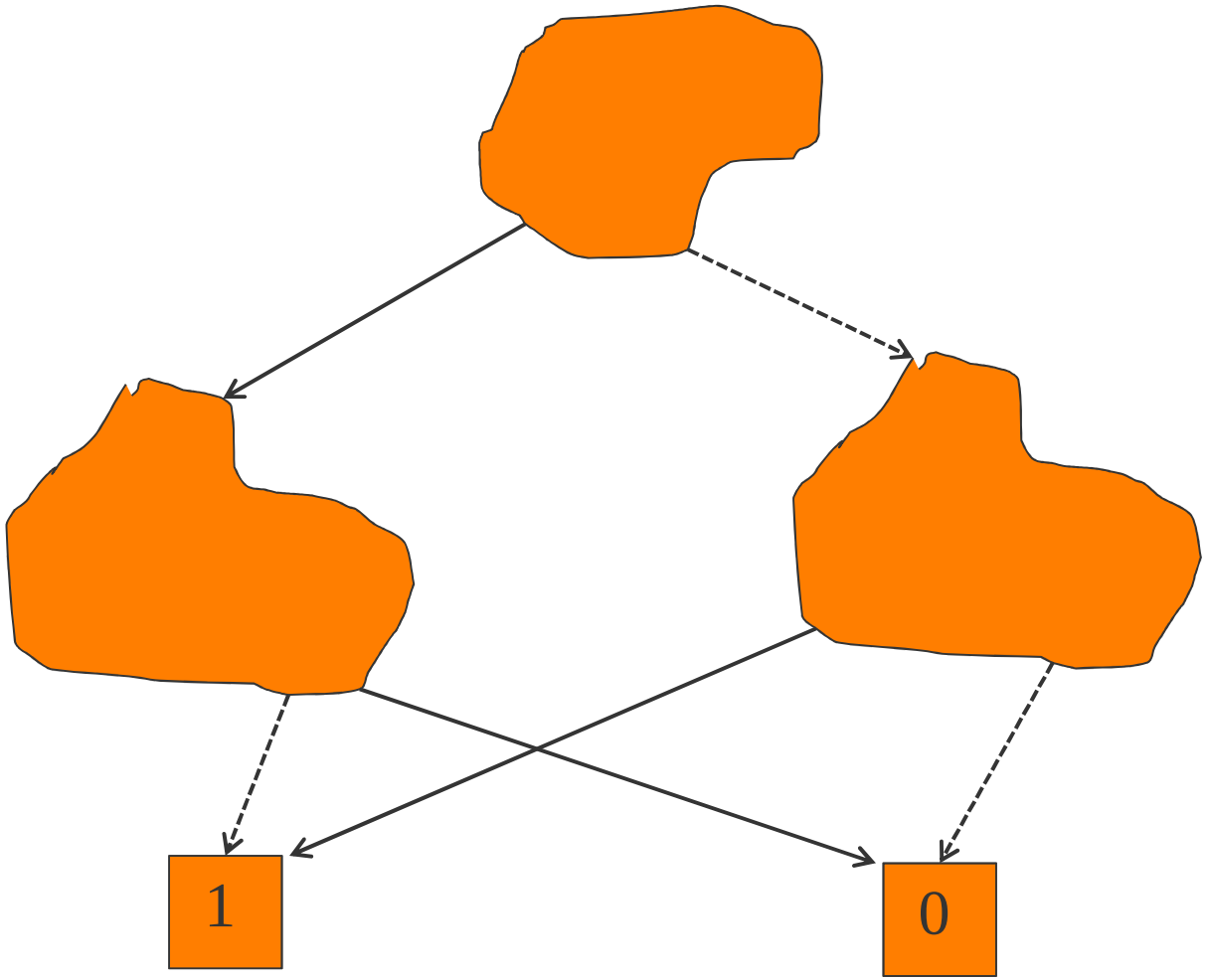
Typische Funktionen

- ODER



Typische Funktionen

- **XOR:** einen Operanden duplizieren



Realisierung als mehrstufige Schaltkreise mittels MUX