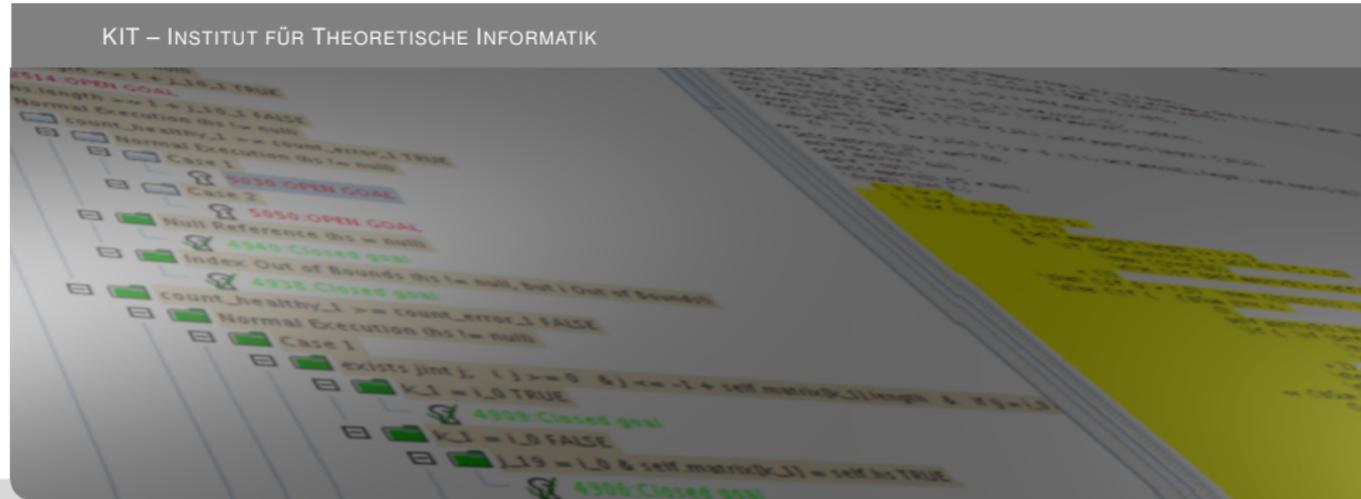


# Formale Systeme

Binary Decision Diagrams  
Prof. Dr. Peter H. Schmitt

KIT – INSTITUT FÜR THEORETISCHE INFORMATIK



# Shannon Formeln

*Shannon Formeln* sind aussagenlogische Formeln, die aufgebaut sind aus

- ▶ dem dreistelligen Operator *sh*

# Shannon Formeln

*Shannon Formeln* sind aussagenlogische Formeln, die aufgebaut sind aus

- ▶ dem dreistelligen Operator  $sh$
- ▶ den Konstanten 0 und 1

# Shannon Formeln

*Shannon Formeln* sind aussagenlogische Formeln, die aufgebaut sind aus

- ▶ dem dreistelligen Operator  $sh$
- ▶ den Konstanten 0 und 1
- ▶ Aussagevariablen  $P_1, \dots, P_n, \dots$

# Shannon Formeln

*Shannon Formeln* sind aussagenlogische Formeln, die aufgebaut sind aus

- ▶ dem dreistelligen Operator  $sh$
- ▶ den Konstanten 0 und 1
- ▶ Aussagevariablen  $P_1, \dots, P_n, \dots$

Semantik von  $sh$  für eine Interpretation  $I$

# Shannon Formeln

*Shannon Formeln* sind aussagenlogische Formeln, die aufgebaut sind aus

- ▶ dem dreistelligen Operator *sh*
- ▶ den Konstanten 0 und 1
- ▶ Aussagevariablen  $P_1, \dots, P_n, \dots$

Semantik von *sh* für eine Interpretation *I*

$$\text{val}_I(\text{sh}(P_1, P_2, P_3)) = \begin{cases} \text{val}_I(P_2) & \text{falls } \text{val}_I(P_1) = F \\ \text{val}_I(P_3) & \text{falls } \text{val}_I(P_1) = W \end{cases}$$

# Shannon Formeln

*Shannon Formeln* sind aussagenlogische Formeln, die aufgebaut sind aus

- ▶ dem dreistelligen Operator *sh*
- ▶ den Konstanten 0 und 1
- ▶ Aussagevariablen  $P_1, \dots, P_n, \dots$

Semantik von *sh* für eine Interpretation *I*

$$\text{val}_I(\text{sh}(P_1, P_2, P_3)) = \begin{cases} \text{val}_I(P_2) & \text{falls } \text{val}_I(P_1) = F \\ \text{val}_I(P_3) & \text{falls } \text{val}_I(P_1) = W \end{cases}$$

oder in Tabellenform:

# Shannon Formeln

*Shannon Formeln* sind aussagenlogische Formeln, die aufgebaut sind aus

- ▶ dem dreistelligen Operator *sh*
- ▶ den Konstanten 0 und 1
- ▶ Aussagevariablen  $P_1, \dots, P_n, \dots$

Semantik von *sh* für eine Interpretation *I*

$$val_I(sh(P_1, P_2, P_3)) = \begin{cases} val_I(P_2) & \text{falls } val_I(P_1) = F \\ val_I(P_3) & \text{falls } val_I(P_1) = W \end{cases}$$

oder in Tabellenform:

|                            |   |   |   |   |   |   |   |   |
|----------------------------|---|---|---|---|---|---|---|---|
| $val_I(P_1)$               | W | W | W | W | F | F | F | F |
| $val_I(P_2)$               | W | W | F | F | W | W | F | F |
| $val_I(P_3)$               | W | F | W | F | W | F | W | F |
| $val_I(sh(P_1, P_2, P_3))$ | W | F | W | F | W | W | F | F |

# Eigenschaften des *sh*-Operators

►  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$
- ▶  $sh(0, P_2, P_3) \leftrightarrow P_2$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$
- ▶  $sh(0, P_2, P_3) \leftrightarrow P_2$
- ▶  $sh(1, P_2, P_3) \leftrightarrow P_3$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$
- ▶  $sh(0, P_2, P_3) \leftrightarrow P_2$
- ▶  $sh(1, P_2, P_3) \leftrightarrow P_3$
- ▶  $sh(P, 0, 1) \leftrightarrow P$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$
- ▶  $sh(0, P_2, P_3) \leftrightarrow P_2$
- ▶  $sh(1, P_2, P_3) \leftrightarrow P_3$
- ▶  $sh(P, 0, 1) \leftrightarrow P$
- ▶  $sh(P, 1, 0) \leftrightarrow \neg P$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$
- ▶  $sh(0, P_2, P_3) \leftrightarrow P_2$
- ▶  $sh(1, P_2, P_3) \leftrightarrow P_3$
- ▶  $sh(P, 0, 1) \leftrightarrow P$
- ▶  $sh(P, 1, 0) \leftrightarrow \neg P$
- ▶  $sh(P_1, P_2, P_2) \leftrightarrow P_2$

# Eigenschaften des *sh*-Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$
- ▶  $sh(0, P_2, P_3) \leftrightarrow P_2$
- ▶  $sh(1, P_2, P_3) \leftrightarrow P_3$
- ▶  $sh(P, 0, 1) \leftrightarrow P$
- ▶  $sh(P, 1, 0) \leftrightarrow \neg P$
- ▶  $sh(P_1, P_2, P_2) \leftrightarrow P_2$
- ▶  $sh(sh(P_1, P_2, P_3), P_4, P_5) \leftrightarrow sh(P_1, sh(P_2, P_4, P_5), sh(P_3, P_4, P_5))$

# Eigenschaften des $sh$ -Operators

- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \vee P_3) \wedge (P_1 \vee P_2)$
- ▶  $sh(P_1, P_2, P_3) \leftrightarrow (P_1 \rightarrow P_3) \wedge (\neg P_1 \rightarrow P_2)$
- ▶  $\neg sh(A, B, C) \leftrightarrow sh(A, \neg B, \neg C)$
- ▶  $sh(0, P_2, P_3) \leftrightarrow P_2$
- ▶  $sh(1, P_2, P_3) \leftrightarrow P_3$
- ▶  $sh(P, 0, 1) \leftrightarrow P$
- ▶  $sh(P, 1, 0) \leftrightarrow \neg P$
- ▶  $sh(P_1, P_2, P_2) \leftrightarrow P_2$
- ▶  $sh(sh(P_1, P_2, P_3), P_4, P_5) \leftrightarrow sh(P_1, sh(P_2, P_4, P_5), sh(P_3, P_4, P_5))$
- ▶  $A \leftrightarrow sh(P, A_{P=0}, A_{P=1})$

Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene.

## Definition

Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene.

## Definition

1. Die Konstanten 0, 1 sind normierte *sh*-Formeln.

Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene.

## Definition

1. Die Konstanten 0, 1 sind normierte *sh*-Formeln.
2.  $sh(P_i, A, B)$  ist eine normierte *sh*-Formel wenn

Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene.

## Definition

1. Die Konstanten 0, 1 sind normierte *sh*-Formeln.
2.  $sh(P_i, A, B)$  ist eine normierte *sh*-Formel wenn
  - ▶  $A$  und  $B$  normierte *sh*-Formeln sind und

Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene.

## Definition

1. Die Konstanten 0, 1 sind normierte *sh*-Formeln.
2.  $sh(P_i, A, B)$  ist eine normierte *sh*-Formel wenn
  - ▶  $A$  und  $B$  normierte *sh*-Formeln sind und
  - ▶ für jede in  $A$  oder  $B$  vorkommende Aussagenvariable  $P_j$  gilt  $j > i$ .

Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene.

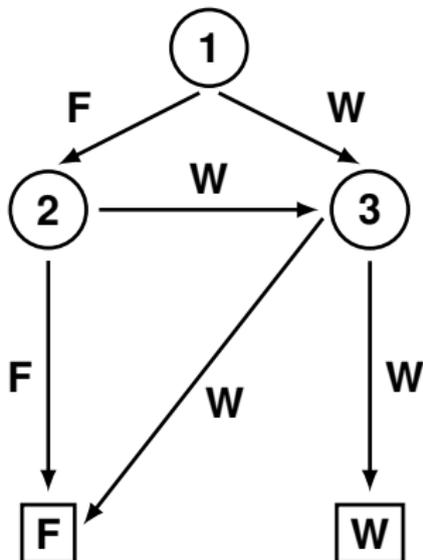
## Definition

1. Die Konstanten 0, 1 sind normierte *sh*-Formeln.
2.  $sh(P_i, A, B)$  ist eine normierte *sh*-Formel wenn
  - ▶  $A$  und  $B$  normierte *sh*-Formeln sind und
  - ▶ für jede in  $A$  oder  $B$  vorkommende Aussagenvariable  $P_j$  gilt  $j > i$ .

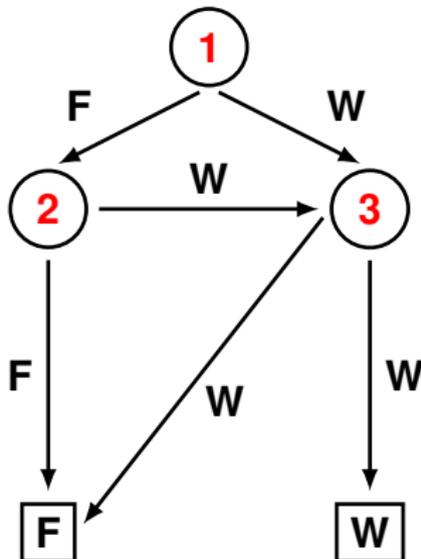
## Theorem

*Zu jeder aussagenlogischen Formel  $A$  gibt es eine äquivalente normierte *sh*-Formel  $B$ .*

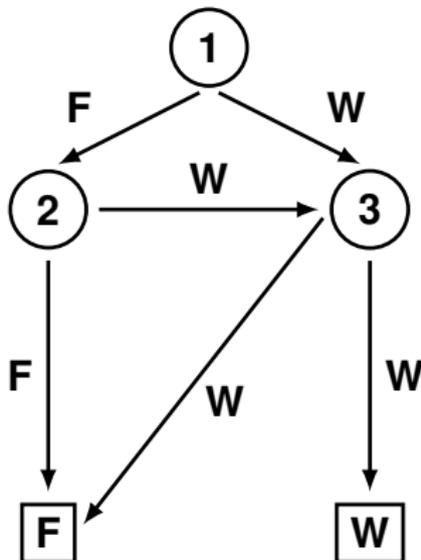
# Shannon Graphen



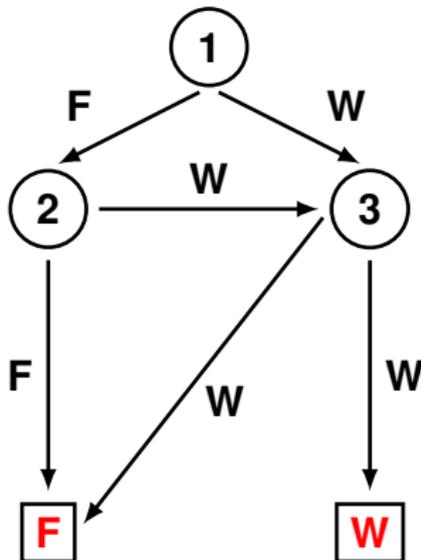
Ein *sh-Graph* ist ein gerichteter, binärer, zusammenhängender Graph.



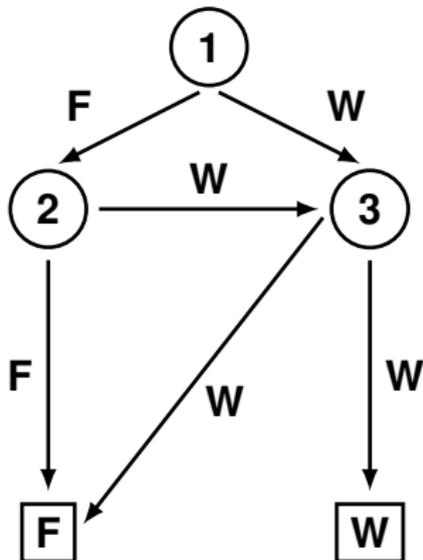
Jedem nichtterminalen Knoten  $v$  ist eine natürliche Zahl  $index(v)$  zugeordnet.



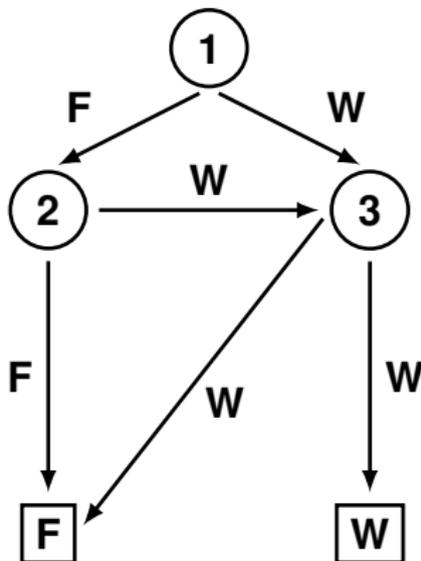
Von jedem nichtterminalen Knoten  $v$  gehen zwei Kanten aus. Eine davon ist mit **F**, die andere mit **W** gekennzeichnet.



Jedem terminalen Knoten  $v$  ist mit **F** oder **W** versehen, bezeichnet mit  $wert(v)$ .

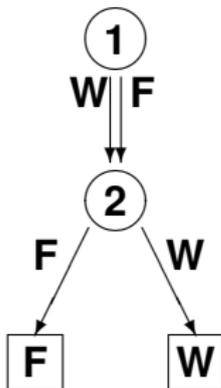
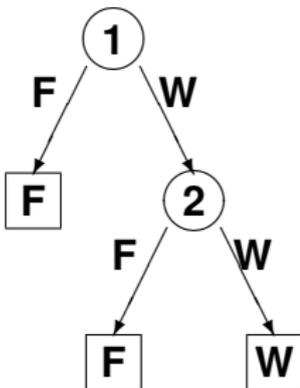
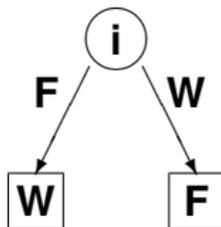
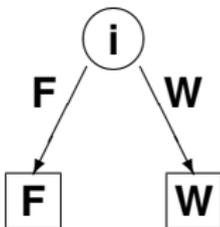


Ist der nichtterminale Knoten  $w$  ein unmittelbarer Nachfolger von  $v$ , dann gilt  $index(v) < index(w)$ .



Es gibt genau einen Wurzelknoten

# Weitere Beispiele von Shannon Graphen



# Shannon Graphen und Boolesche Funktionen

- ▶ Jedem  $sh$ -Graphen  $G$  kann man eine  $m$ -stellige Boolesche Funktion  $f_G$  zuordnen, wobei  $m$  die Anzahl der in  $G$  vorkommenden verschiedenen Indizes  $i_1, \dots, i_m$  ist.

# Shannon Graphen und Boolesche Funktionen

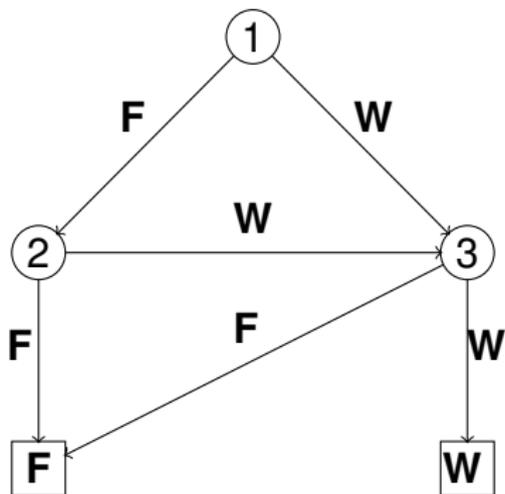
- ▶ Jedem  $sh$ -Graphen  $G$  kann man eine  $m$ -stellige Boolesche Funktion  $f_G$  zuordnen, wobei  $m$  die Anzahl der in  $G$  vorkommenden verschiedenen Indizes  $i_1, \dots, i_m$  ist.
- ▶ Wir fassen  $f_G$  als eine Funktion mit den Eingabevariablen  $P_{i_1}, \dots, P_{i_m}$  auf und bestimmen den Funktionswert  $f_G(P_{i_1}, \dots, P_{i_m})$ , indem wir an der Wurzel von  $G$  beginnend einen Pfad durch  $G$  wählen. Am Knoten  $v$  folgen wir der Kante **F**, wenn die Eingabevariable  $P_{index(v)}$  den Wert  $F$  hat, sonst der Kante **W**.

# Shannon Graphen und Boolesche Funktionen

- ▶ Jedem  $sh$ -Graphen  $G$  kann man eine  $m$ -stellige Boolesche Funktion  $f_G$  zuordnen, wobei  $m$  die Anzahl der in  $G$  vorkommenden verschiedenen Indizes  $i_1, \dots, i_m$  ist.
- ▶ Wir fassen  $f_G$  als eine Funktion mit den Eingabevariablen  $P_{i_1}, \dots, P_{i_m}$  auf und bestimmen den Funktionswert  $f_G(P_{i_1}, \dots, P_{i_m})$ , indem wir an der Wurzel von  $G$  beginnend einen Pfad durch  $G$  wählen. Am Knoten  $v$  folgen wir der Kante **F**, wenn die Eingabevariable  $P_{index(v)}$  den Wert  $F$  hat, sonst der Kante **W**.
- ▶ Der Wert des terminalen Knotens ist der gesuchte Funktionswert.

# Shannongraph als Boolesche Funktion

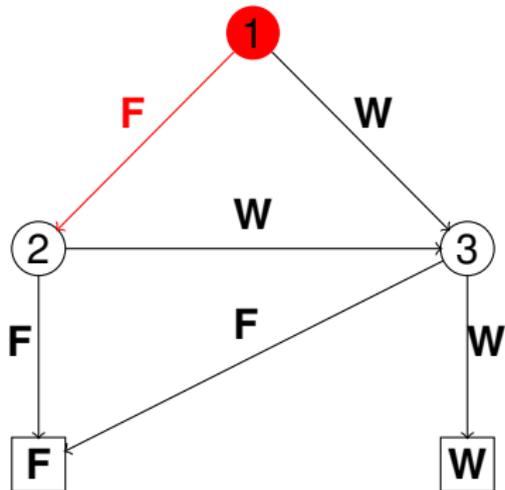
G:



$$f_G(F, W, F) = ?$$

# Shannongraph als Boolesche Funktion

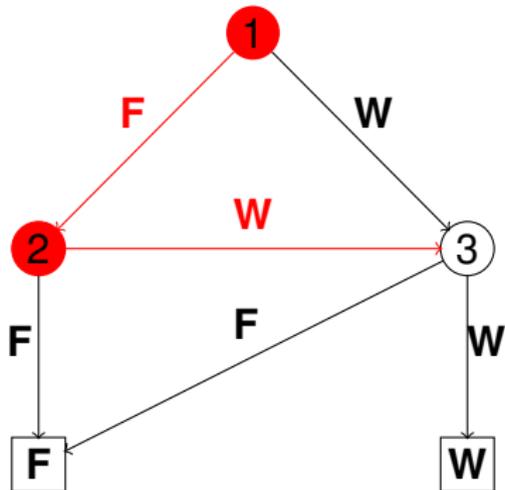
G:



$$f_G(F, W, F) = ?$$

# Shannongraph als Boolesche Funktion

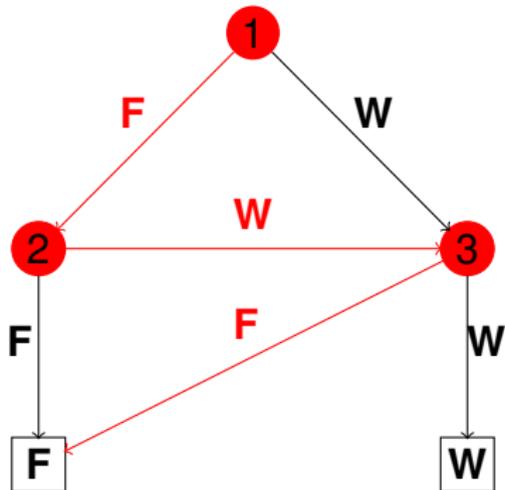
G:



$$f_G(F, W, F) = ?$$

# Shannongraph als Boolesche Funktion

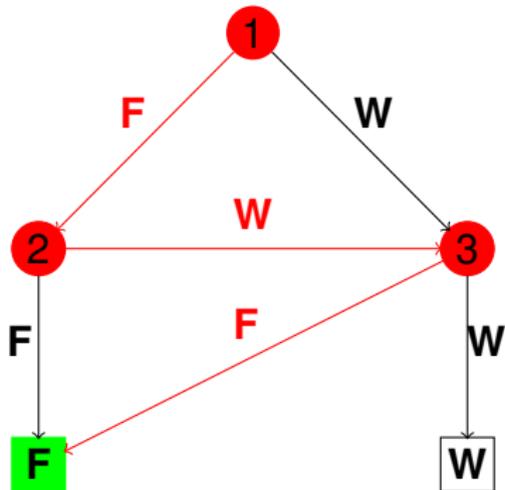
G:



$$f_G(F, W, F) = ?$$

# Shannongraph als Boolesche Funktion

G:



$$f_G(F, W, F) = F$$

# Konstruktion von Shannon Graphen

## Beispiel

$P_1$

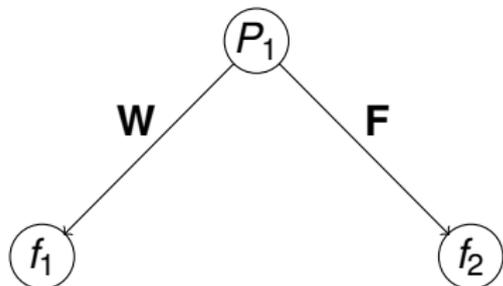
$f(P_1, P_2, P_3)$

$$\begin{cases} W & \text{card}(\{1 \leq i \leq 3 \mid P_i = W\}) = 2 \\ F & \text{sonst} \end{cases}$$

=

# Konstruktion von Shannon Graphen

Beispiel



$$f(P_1, P_2, P_3)$$

$$\begin{cases} W & \text{card}(\{1 \leq i \leq 3 \mid P_i = W\}) = 2 \\ F & \text{sonst} \end{cases}$$

$$f_1(P_2, P_3) = f(W, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } (P_2 = W \text{ und } P_3 = F) \text{ oder} \\ & (P_2 = F \text{ und } P_3 = W) \\ F & \text{sonst} \end{cases}$$

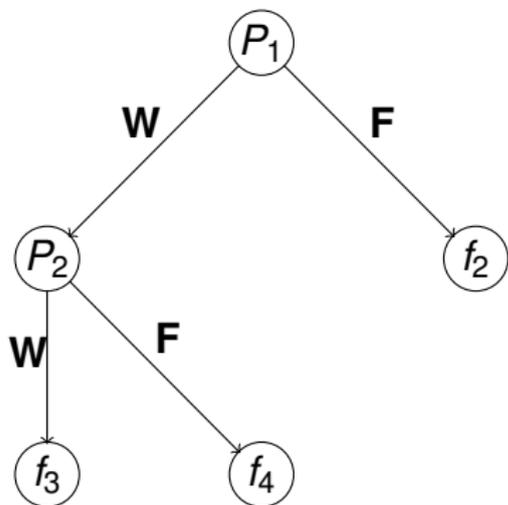
$$f_2(P_2, P_3) = f(F, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } P_2 = P_3 = W \\ F & \text{sonst} \end{cases}$$

=

# Konstruktion von Shannon Graphen

Beispiel



$$f(P_1, P_2, P_3)$$

=

$$\begin{cases} W & \text{card}(\{1 \leq i \leq 3 \mid P_i = W\}) = 2 \\ F & \text{sonst} \end{cases}$$

$$f_1(P_2, P_3) = f(W, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } (P_2 = W \text{ und } P_3 = F) \text{ oder} \\ & (P_2 = F \text{ und } P_3 = W) \\ F & \text{sonst} \end{cases}$$

$$f_2(P_2, P_3) = f(F, P_2, P_3) =$$

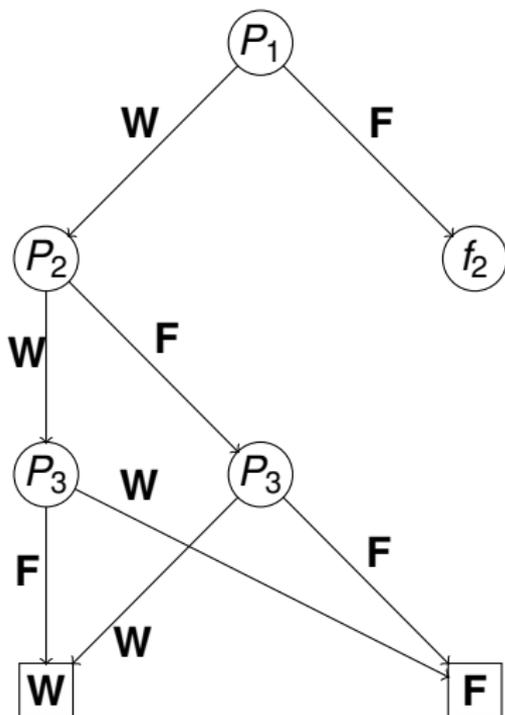
$$\begin{cases} W & \text{falls } P_2 = P_3 = W \\ F & \text{sonst} \end{cases}$$

$$f_3(P_3) = f(W, W, P_3) = \neg P_3$$

$$f_4(P_3) = f(W, F, P_3) = P_3$$

# Konstruktion von Shannon Graphen

## Beispiel



$$f(P_1, P_2, P_3)$$

=

$$\begin{cases} W & \text{card}(\{1 \leq i \leq 3 \mid P_i = W\}) = 2 \\ F & \text{sonst} \end{cases}$$

$$f_1(P_2, P_3) = f(W, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } (P_2 = W \text{ und } P_3 = F) \text{ oder} \\ & (P_2 = F \text{ und } P_3 = W) \\ F & \text{sonst} \end{cases}$$

$$f_2(P_2, P_3) = f(F, P_2, P_3) =$$

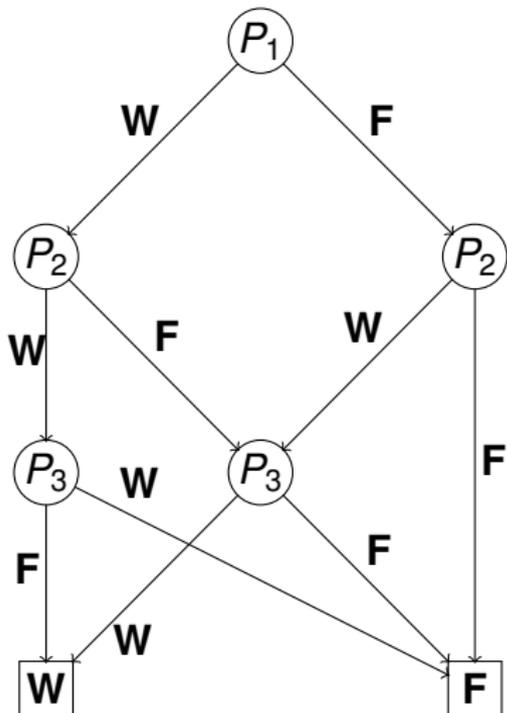
$$\begin{cases} W & \text{falls } P_2 = P_3 = W \\ F & \text{sonst} \end{cases}$$

$$f_3(P_3) = f(W, W, P_3) = \neg P_3$$

$$f_4(P_3) = f(W, F, P_3) = P_3$$

# Konstruktion von Shannon Graphen

## Beispiel



$$f(P_1, P_2, P_3)$$

=

$$\begin{cases} W & \text{card}(\{1 \leq i \leq 3 \mid P_i = W\}) = 2 \\ F & \text{sonst} \end{cases}$$

$$f_1(P_2, P_3) = f(W, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } (P_2 = W \text{ und } P_3 = F) \text{ oder} \\ & (P_2 = F \text{ und } P_3 = W) \\ F & \text{sonst} \end{cases}$$

$$f_2(P_2, P_3) = f(F, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } P_2 = P_3 = W \\ F & \text{sonst} \end{cases}$$

$$f_3(P_3) = f(W, W, P_3) = \neg P_3$$

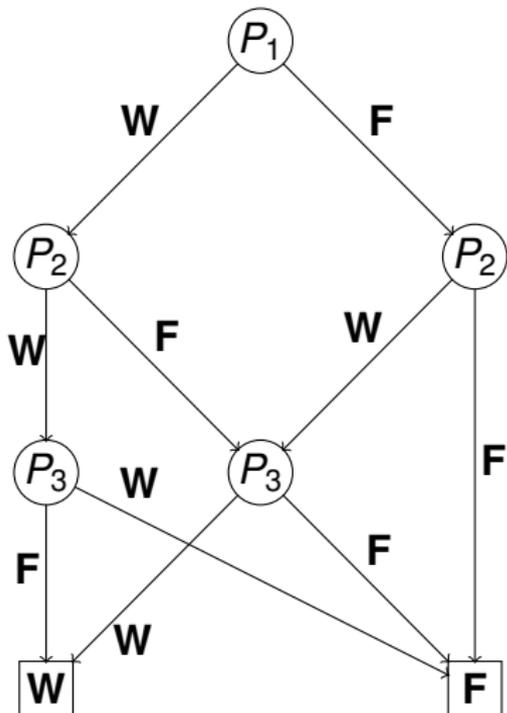
$$f_4(P_3) = f(W, F, P_3) = P_3$$

$$f_5(P_3) = f(F, F, P_3) = F$$

$$f_6(P_3) = f(F, W, P_3) = P_3$$

# Konstruktion von Shannon Graphen

## Beispiel



$$f(P_1, P_2, P_3)$$

=

$$\begin{cases} W & \text{card}(\{1 \leq i \leq 3 \mid P_i = W\}) = 2 \\ F & \text{sonst} \end{cases}$$

$$f_1(P_2, P_3) = f(W, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } (P_2 = W \text{ und } P_3 = F) \text{ oder} \\ & (P_2 = F \text{ und } P_3 = W) \\ F & \text{sonst} \end{cases}$$

$$f_2(P_2, P_3) = f(F, P_2, P_3) =$$

$$\begin{cases} W & \text{falls } P_2 = P_3 = W \\ F & \text{sonst} \end{cases}$$

$$f_3(P_3) = f(W, W, P_3) = \neg P_3$$

$$f_4(P_3) = f(W, F, P_3) = P_3$$

$$f_5(P_3) = f(F, F, P_3) = F$$

$$f_6(P_3) = f(F, W, P_3) = P_3$$

card(X) = Anzahl der Element in der Menge X

# Shannon Graphen vs normierte Shannon Formeln

Es gibt eine offensichtliche Korrespondenz  
zwischen

Shannon Graphen

und

normierten Shannon Formeln:

*n*-te Variable entspricht Knoten mit Index *n*

Von jetzt an betrachten wir nur noch Shannon Graphen.

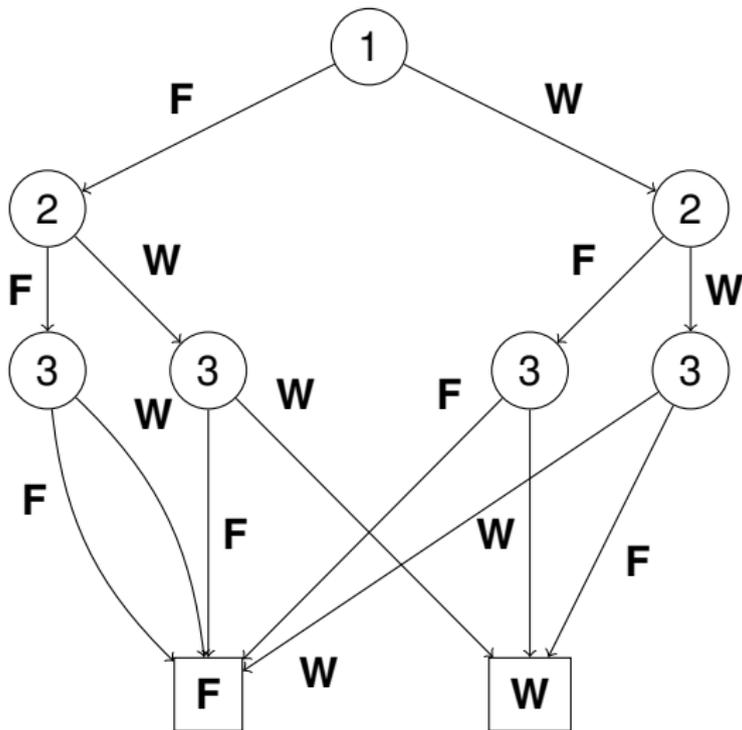
## Definition

Ein *sh*-Graph heißt *reduziert*, wenn

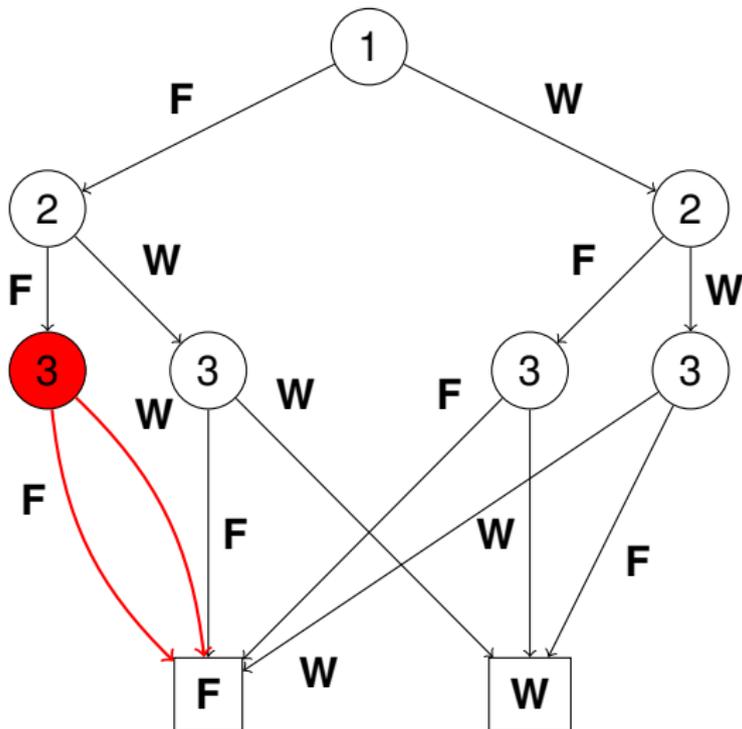
1. es keine zwei Knoten  $v$  und  $w$  ( $v \neq w$ ) gibt, so daß der in  $v$  verwurzelte Teilgraph  $G_v$  mit dem in  $w$  verwurzelten Teilgraph  $G_w$  isomorph ist.
2. es keinen Knoten  $v$  gibt, so dass die beiden von  $v$  ausgehenden Kanten zum selben Nachfolgerknoten führen.

Ein reduzierter Shannongraph heißt auch *ordered binary decision diagram* (OBDD).

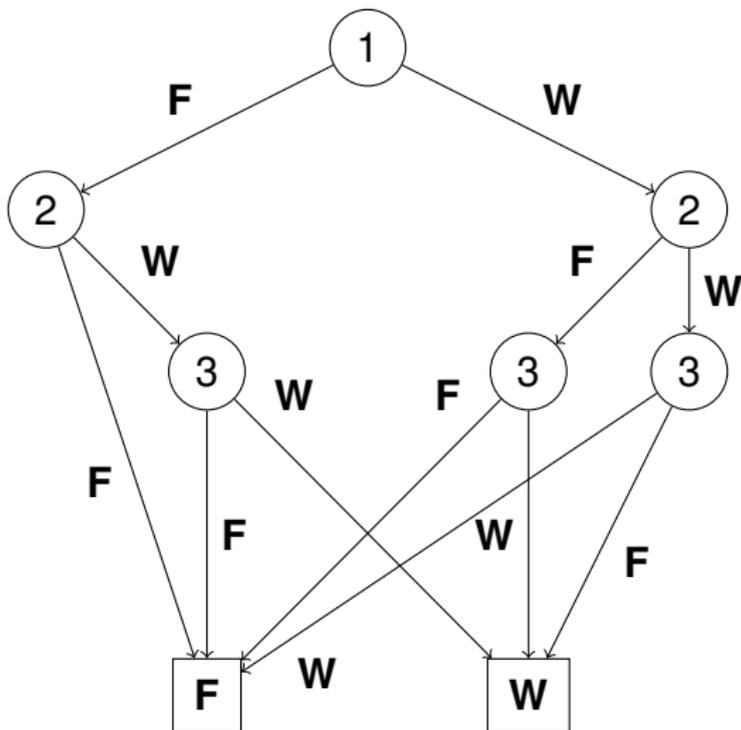
# Ein Beispiel



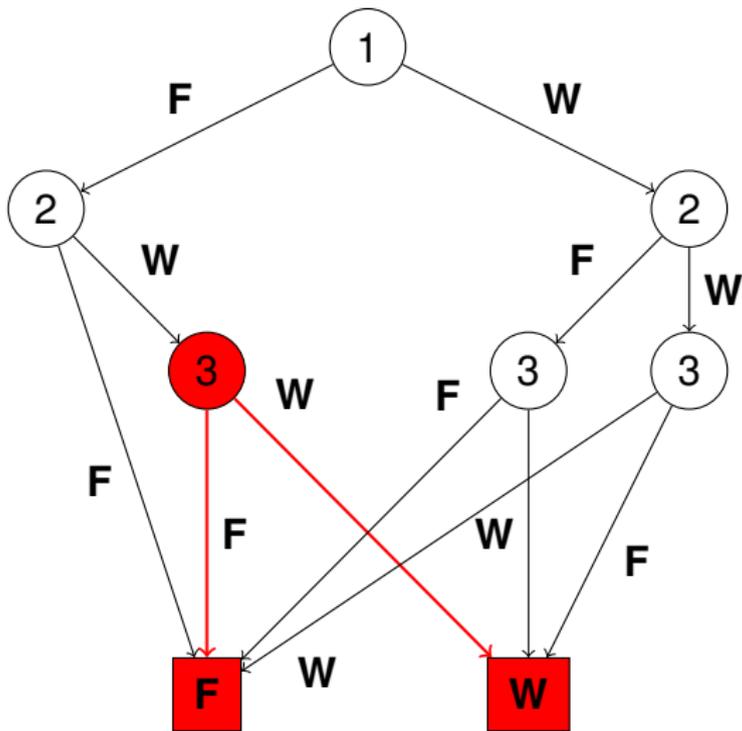
# Doppelte Kanten



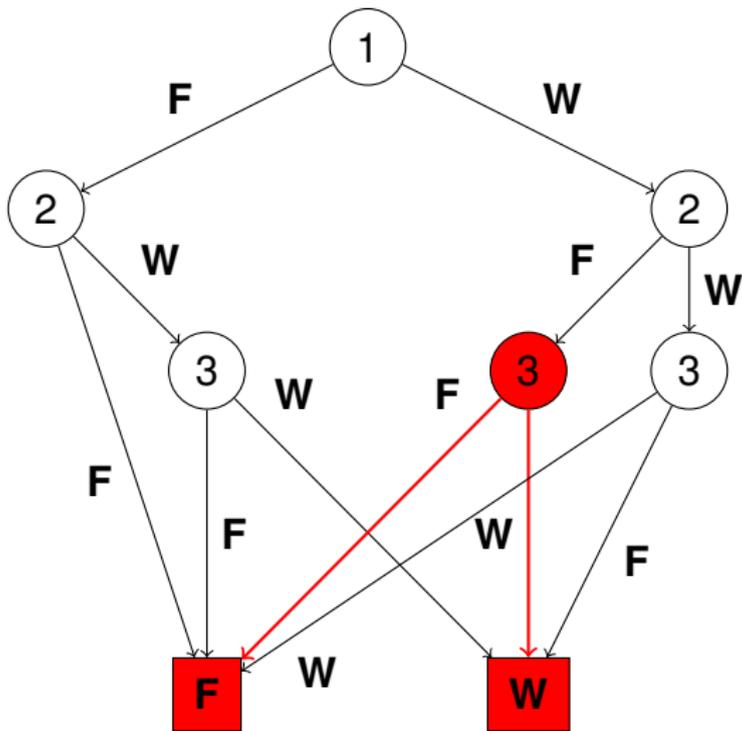
# Elimination doppelter Kanten



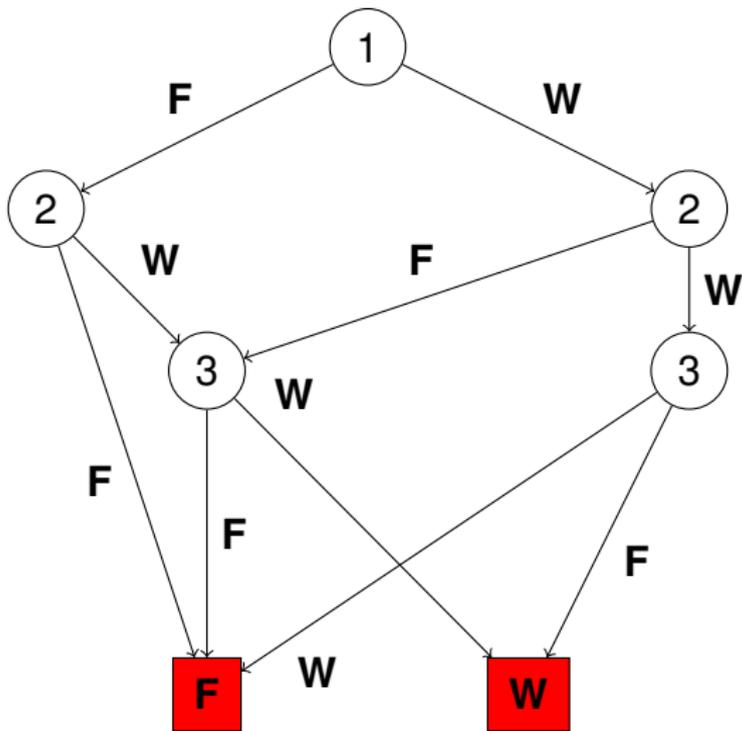
# Isomorphe Teilgraphen



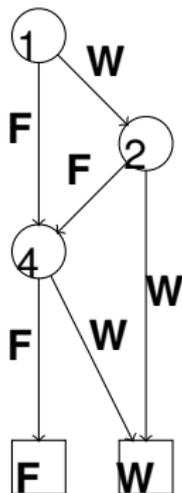
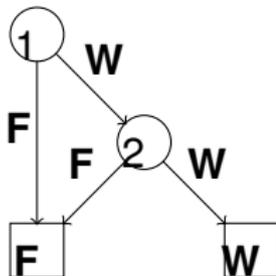
# Isomorphe Teilgraphen



# Reduktion isomorpher Teilgraphen



# Weitere Beispiele



## Definition

Seien zwei *sh*-Graphen  $H, G$  gegeben. Ihre Knotenmengen seien  $V_1, V_2$ .

$H, G$  heißen zueinander *isomorph* ( $H \cong G$ ) genau dann, wenn es eine bijektive Abbildung  $\pi$  von  $V_1$  nach  $V_2$  gibt mit:

## Definition

Seien zwei *sh*-Graphen  $H, G$  gegeben. Ihre Knotenmengen seien  $V_1, V_2$ .

$H, G$  heißen zueinander *isomorph* ( $H \cong G$ ) genau dann, wenn es eine bijektive Abbildung  $\pi$  von  $V_1$  nach  $V_2$  gibt mit:

1.  $index(k) = index(\pi(k))$  für jeden Nichtterminalknoten  $k \in V_1$

## Definition

Seien zwei *sh*-Graphen  $H, G$  gegeben. Ihre Knotenmengen seien  $V_1, V_2$ .

$H, G$  heißen zueinander *isomorph* ( $H \cong G$ ) genau dann, wenn es eine bijektive Abbildung  $\pi$  von  $V_1$  nach  $V_2$  gibt mit:

1.  $index(k) = index(\pi(k))$  für jeden Nichtterminalknoten  $k \in V_1$
2.  $wert(k) = wert(\pi(k))$  für jeden Terminalknoten  $k \in V_1$

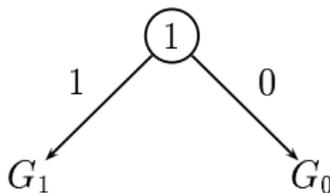
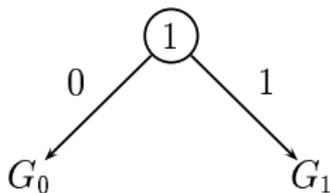
## Definition

Seien zwei *sh*-Graphen  $H, G$  gegeben. Ihre Knotenmengen seien  $V_1, V_2$ .

$H, G$  heißen zueinander *isomorph* ( $H \cong G$ ) genau dann, wenn es eine bijektive Abbildung  $\pi$  von  $V_1$  nach  $V_2$  gibt mit:

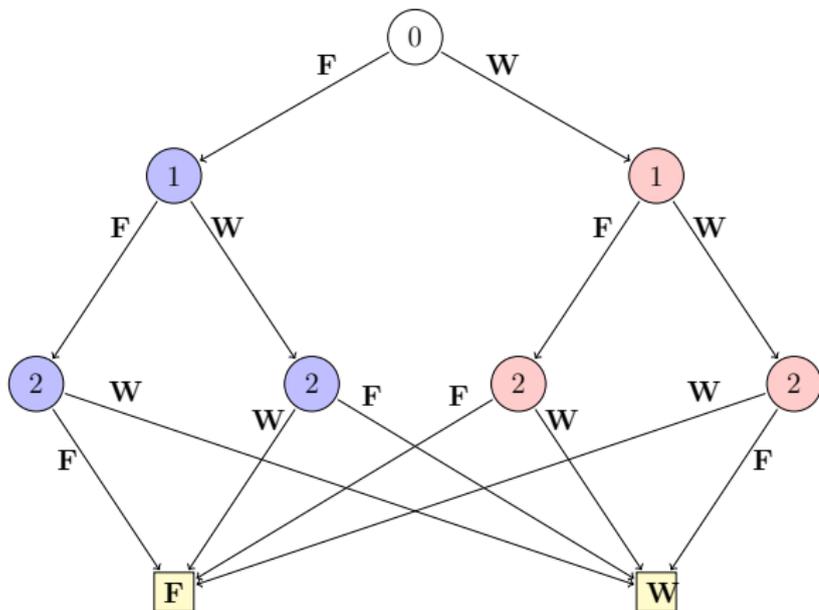
1.  $index(k) = index(\pi(k))$  für jeden Nichtterminalknoten  $k \in V_1$
2.  $wert(k) = wert(\pi(k))$  für jeden Terminalknoten  $k \in V_1$
3. Für jeden Nichtterminalknoten  $k \in V_1$ , dessen **F**-Kante/**W**-Kante zu dem Knoten  $k_0/k_1$  führt, gilt: die **F**-Kante von  $\pi(k)$  führt zu  $\pi(k_0)$ , die **W**-Kante zu  $\pi(k_1)$ .

# Einfachstes Beispiel isomorpher Shannon Graphen

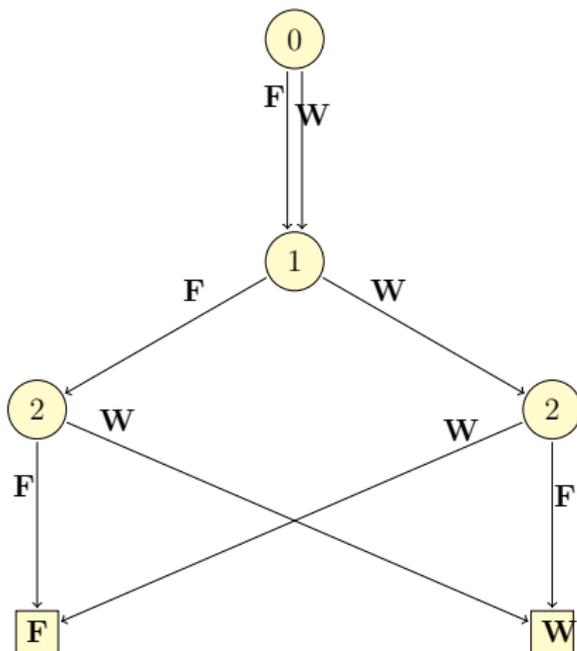


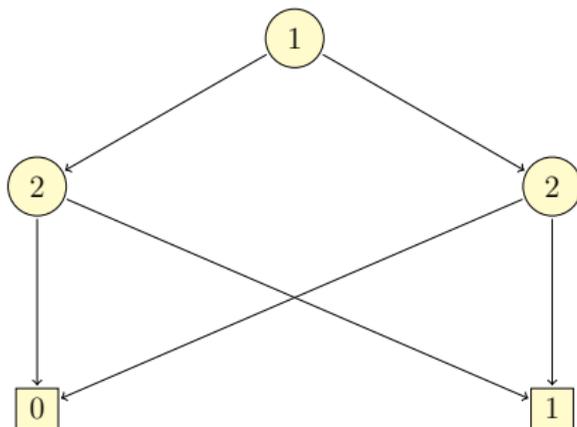
# Komplexeres Beispiel

Isomorphismus  $\pi$  : 



# Reduktion bezgl $\pi$





## Theorem

*Sei  $G$  ein Shannongraph, so daß für jedes Paar von Knoten  $v, w$  gilt*

*wenn die **W**-Nachfolger von  $v$  und  $w$  gleich sind und  
die **F**-Nachfolger von  $v$  und  $w$  gleich sind*

*dann  $v = w$*

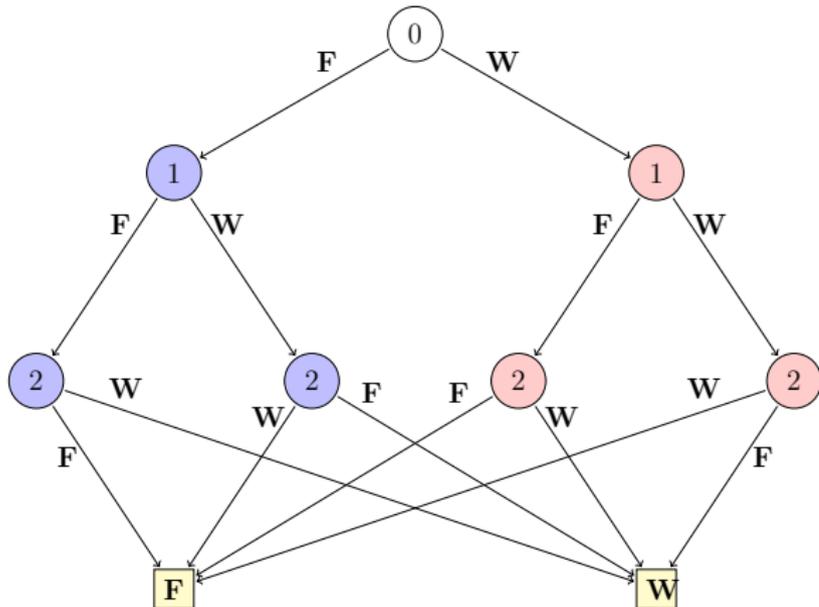
*Dann erfüllt  $G$  die Bedingung (1) aus der Definition reduzierter Shannongraphen, d.h. für jedes Paar  $x, y$  von Knoten gilt*

*wenn  $G_x$  isomorph zu  $G_y$  ist*

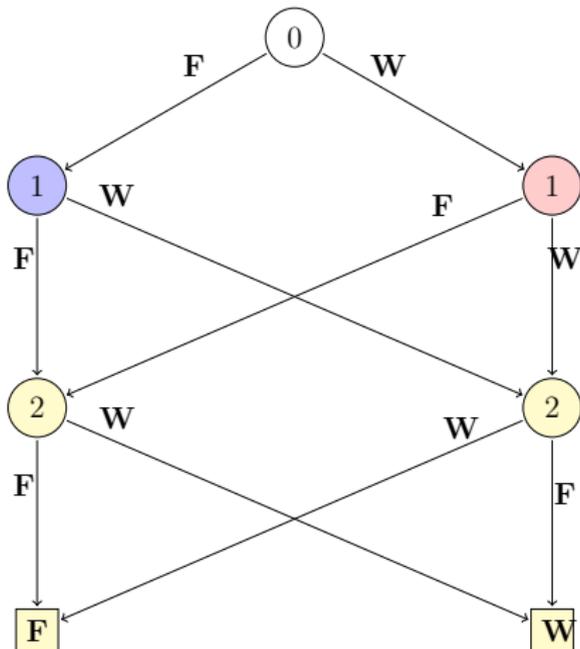
*dann  $x = y$*

# Komplexeres Beispiel (nochmal)

Isomorphismus  $\pi$  : 



# Inkrementelle Reduktion



# Eindeutigkeit reduzierter Shannon Graphen

## Theorem

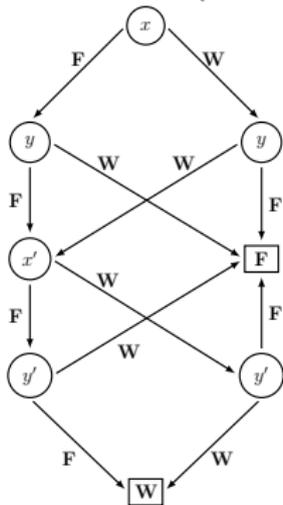
*Sind  $G, H$  reduzierte sh-Graphen zu  $\Sigma = \{P_1, \dots, P_n\}$ , dann gilt*

$$f_G = f_H \Leftrightarrow G \cong H.$$

*(Zu jeder Booleschen Funktion  $f$  gibt es bis auf Isomorphie genau einen reduzierten sh-Graphen  $H$  mit  $f = f_H$ ).*

# Abhängigkeit von der Variablenordnung

Zwei BDDs für  $(x \leftrightarrow y) \wedge (x' \leftrightarrow y')$

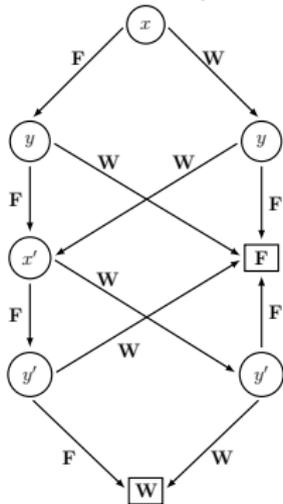


Ordnung:

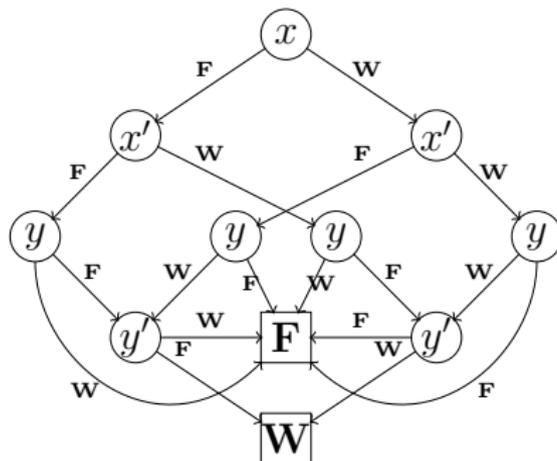
$x < y < x' < y'$

# Abhängigkeit von der Variablenordnung

Zwei BDDs für  $(x \leftrightarrow y) \wedge (x' \leftrightarrow y')$



Ordnung:  
 $x < y < x' < y'$



Ordnung:  
 $x < x' < y < y'$

[BDD für Multiplikationen]

- ▶  $X$  enthalte  $2k$  Variablen  $\{x_0, \dots, x_{k-1}, y_0, \dots, y_{k-1}\}$

[BDD für Multiplikationen]

- ▶  $X$  enthalte  $2k$  Variablen  $\{x_0, \dots, x_{k-1}, y_0, \dots, y_{k-1}\}$
- ▶  $x = x_0 \dots x_{k-1}$  und  $y = y_0 \dots y_{k-1}$  bezeichnen  $k$ -stellige Binärzahlen.

[BDD für Multiplikationen]

- ▶  $X$  enthalte  $2k$  Variablen  $\{x_0, \dots, x_{k-1}, y_0, \dots, y_{k-1}\}$
- ▶  $x = x_0 \dots x_{k-1}$  und  $y = y_0 \dots y_{k-1}$  bezeichnen  $k$ -stellige Binärzahlen.
- ▶ für  $0 \leq i < 2k$  bezeichne  $Mult_i$  die boolesche Funktion, die das  $i$ -te Bit des Produktes von  $x$  mit  $y$  beschreibt.

[BDD für Multiplikationen]

- ▶  $X$  enthalte  $2k$  Variablen  $\{x_0, \dots, x_{k-1}, y_0, \dots, y_{k-1}\}$
- ▶  $x = x_0 \dots x_{k-1}$  und  $y = y_0 \dots y_{k-1}$  bezeichnen  $k$ -stellige Binärzahlen.
- ▶ für  $0 \leq i < 2k$  bezeichne  $Mult_i$  die boolesche Funktion, die das  $i$ -te Bit des Produktes von  $x$  mit  $y$  beschreibt.

[BDD für Multiplikationen]

- ▶  $X$  enthalte  $2k$  Variablen  $\{x_0, \dots, x_{k-1}, y_0, \dots, y_{k-1}\}$
- ▶  $x = x_0 \dots x_{k-1}$  und  $y = y_0 \dots y_{k-1}$  bezeichnen  $k$ -stellige Binärzahlen.
- ▶ für  $0 \leq i < 2k$  bezeichne  $Mult_i$  die boolesche Funktion, die das  $i$ -te Bit des Produktes von  $x$  mit  $y$  beschreibt.

## Theorem

*Für jede Ordnung  $<$  der Variablen in  $X$  gibt es einen Index  $0 \leq i < 2k$ , so dass der BDD  $B_{Mult_i, <}$  mindestens  $2^{k/8}$  Knoten besitzt.*