

***Vorlesung***  
***– Automatisches Beweisen –***  
***Kap. 4.2.4: Binäre Entscheidungsdiagramme (BDD)***

**Prof. Dr. Wolfgang Küchlin**

*Dipl.-Inform., Dr. sc. techn. (ETH)*

**Arbeitsbereich Symbolisches Rechnen  
Wilhelm-Schickard-Institut für Informatik  
Fakultät für Informations- und Kognitionswissenschaften**

**Universität Tübingen**

**Steinbeis Transferzentrum  
Objekt- und Internet-Technologien (OIT)**

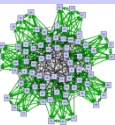
**[Wolfgang.Kuechlin@uni-tuebingen.de](mailto:Wolfgang.Kuechlin@uni-tuebingen.de)  
<http://www-sr.informatik.uni-tuebingen.de>**



# Binäre Entscheidungsdiagramme (BDDs)

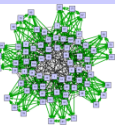
---

- Ursprünglich von Lee und Akers vorgeschlagen
  - C. Y. Lee. Representation of Switching Circuits by Binary-Decision Programs. *Bell Systems Tech. J.*, **38**:985–999, 1959
  - Sheldon B. Akers. Binary Decision Diagrams, *IEEE Transactions on Computers*, **C-27**(6):509–516, Juni 1978.
- Graphen-basierter Formalismus zur Darstellung boolescher Funktionen
- Aussagenlogische Formeln repräsentiert als gerichteter, azyklischer Graph (DAG)
- Weiterentwickelt 1986 von R. Bryant (OBDD, ROBDD)
  - Randal E. Bryant: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Computers*, Vol. C-35, No. 8 (Aug 1986), 677-691.
- Liefern als ROBDDs eine **kanonische** Repräsentierung von Formeln (eindeutig in jeder Äquivalenzklasse).



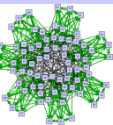
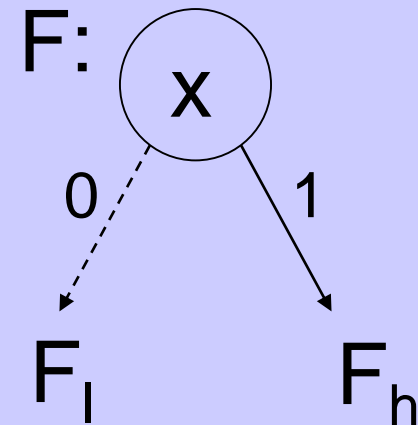
# ROBDDs: Canonicity

- ROBDD  $b$  zu gegebener booleschen Funktion  $f$  ist eindeutig, d.h.:
  - $b$  ist eindeutiger Repräsentant aller zu  $f$  äquivalenten Formeln (kanonische Form).
  - Spezialfall: Unerfüllbare (allgemeingültige) Formeln werden durch 0-Knoten (1-Knoten) repräsentiert.
- Konsequenzen:
  - $F \equiv G$  gdw.  $\text{ROBDD}(F) = \text{ROBDD}(G)$
  - $F \equiv \top$  gdw.  $\text{ROBDD}(F) = \text{ROBDD}(\top) = \boxed{1}$
  - $F \equiv \perp$  gdw.  $\text{ROBDD}(F) = \text{ROBDD}(\perp) = \boxed{0}$
  - $F$  erfüllbar gdw.  $\text{ROBDD}(F) \neq \boxed{0}$
  - SAT-Problem auf ROBDDs in  $O(1)$  lösbar



# Binäre Entscheidungsdiagramme (BDDs)

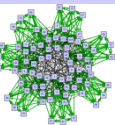
- Terminalknoten  $\boxed{0}$  und  $\boxed{1}$  stellen konstante Funktionen  $\perp$  bzw.  $\top$  dar.
- Innere Knoten interpretiert als:  
 $F = \text{if } x \text{ then } F_h \text{ else } F_l$   
 $(x \Rightarrow F_h) \wedge (\neg x \Rightarrow F_l)$



# BDDs: Bestimmung von $F_h$ und $F_l$

➤ Die **Restriktion**  $F|_{x=b}$  ( $b \in \{0,1\}$ ) ist rekursiv definiert durch:

- $\perp|_{x=b} = \perp$  und  $\top|_{x=b} = \top$
- $y|_{x=b} = \begin{cases} \top & \text{falls } x = y \text{ und } b = 1 \\ \perp & \text{falls } x = y \text{ und } b = 0 \\ y & \text{sonst} \end{cases}$
- $(\neg G)|_{x=b} = \neg(G|_{x=b})$
- $(G \vee H)|_{x=b} = G|_{x=b} \vee H|_{x=b}$
- $(G \wedge H)|_{x=b} = G|_{x=b} \wedge H|_{x=b}$



# BDDs: Bestimmung von $F_h$ und $F_l$ (2)

## ➤ Shannon-Expansion:

$$F \equiv (x \wedge F) \vee (\neg x \wedge F)$$

$$\equiv (x \wedge F|_{x=1}) \vee (\neg x \wedge F|_{x=0})$$

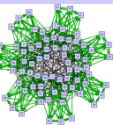
$$\equiv ((x \wedge F|_{x=1}) \vee \neg x) \wedge ((x \wedge F|_{x=1}) \vee F|_{x=0})$$

$$\equiv ((x \vee \neg x) \wedge (F|_{x=1} \vee \neg x)) \wedge ((x \vee F|_{x=0}) \wedge (F|_{x=1} \vee F|_{x=0}))$$

$$\equiv (\neg x \vee F|_{x=1}) \wedge (x \vee F|_{x=0}) \quad // \text{ denn } (F|_{x=1} \vee F|_{x=0}) \text{ ist Resolvente hiervon}$$

$$\equiv (x \Rightarrow F|_{x=1}) \wedge (\neg x \Rightarrow F|_{x=0})$$

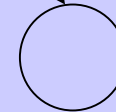
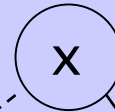
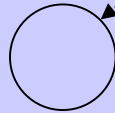
$$\rightarrow F_l \triangleq F|_{x=0} \text{ und } F_h \triangleq F|_{x=1}$$



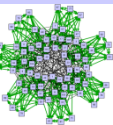
# BDDs: Beispiel zur Generierung

➤  $F = (x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg z$

$F|_{x=0} = y \wedge (\neg y \vee z) \wedge \neg z$

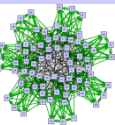
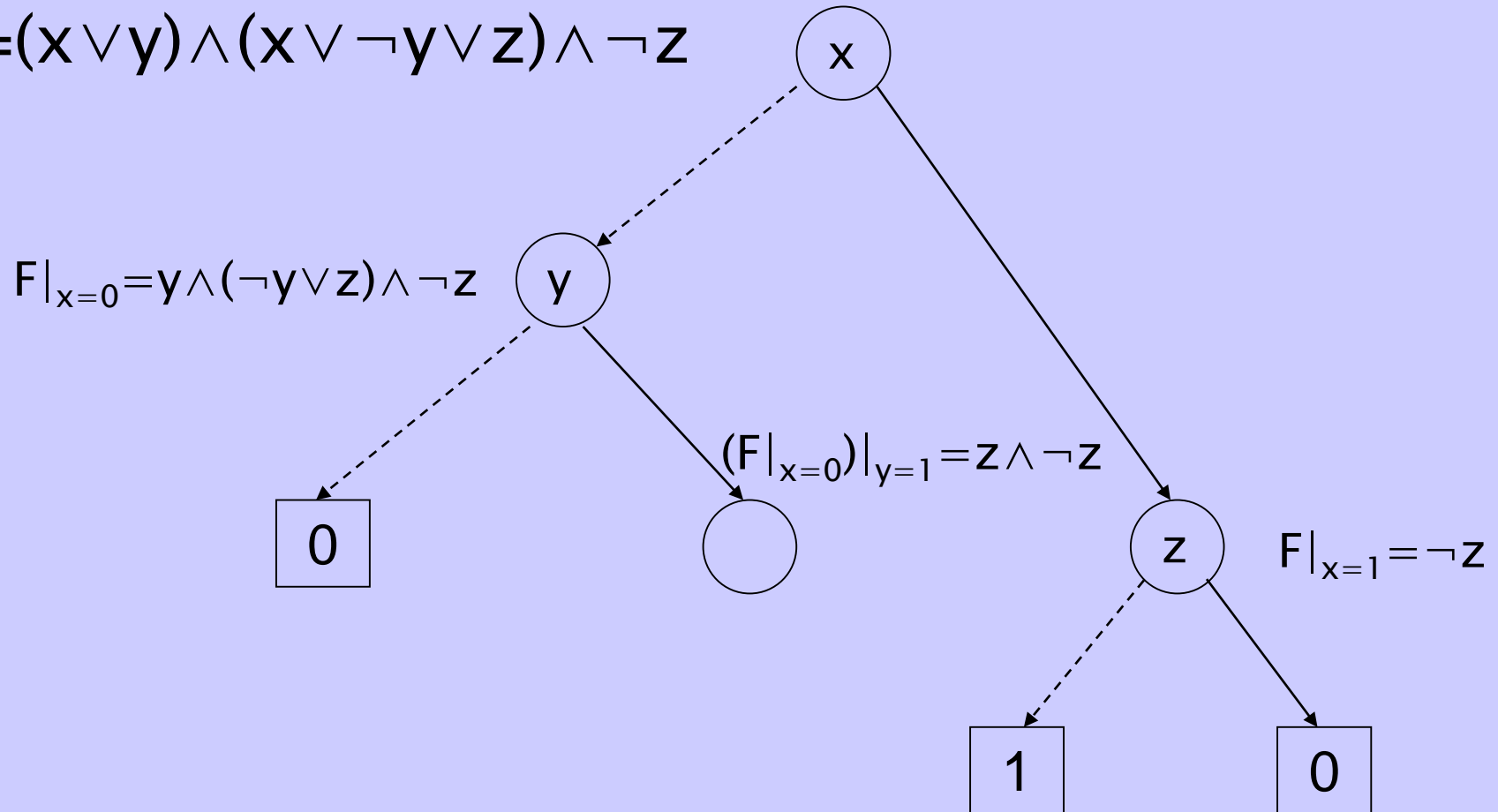


$F|_{x=1} = \neg z$



# BDDs: Beispiel zur Generierung

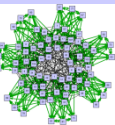
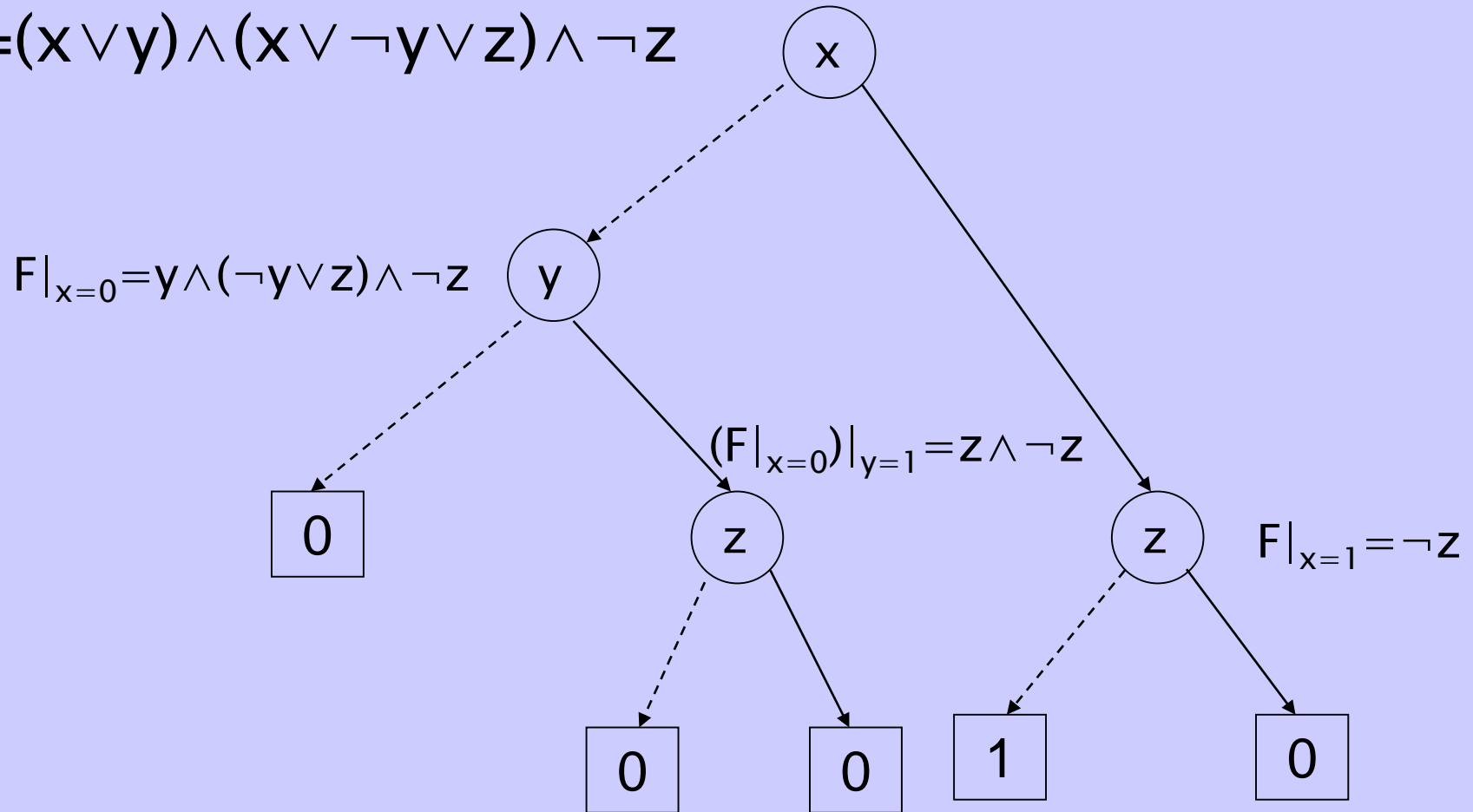
➤  $F = (x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg z$





# BDDs: Beispiel zur Generierung

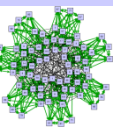
➤  $F = (x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg z$



# Reduced Ordered BDDs (ROBDDs)

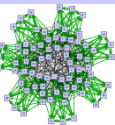
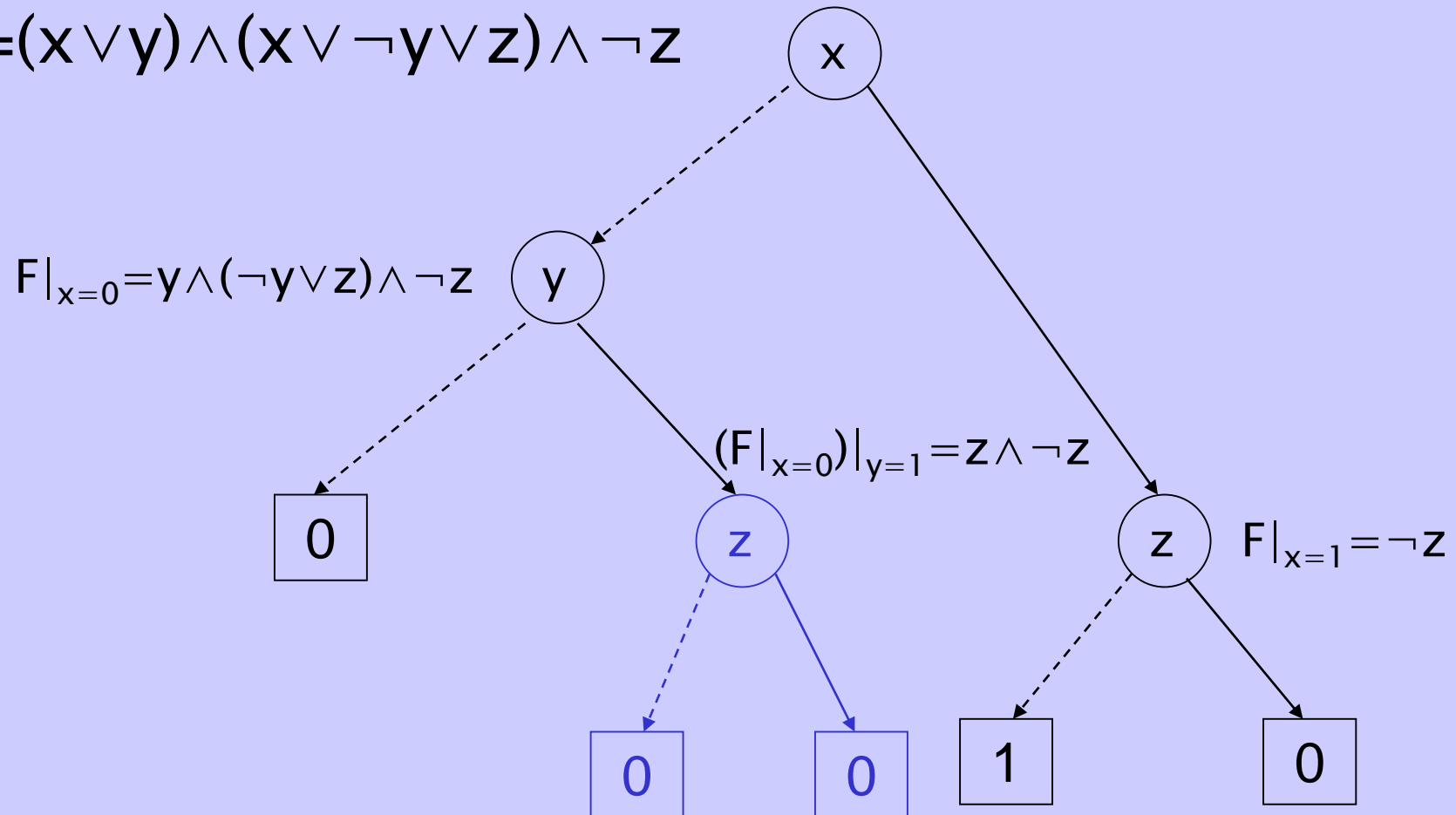
---

- strikte, totale Ordnung  $<$  auf Variablen
  - Variable des Elternknotens muss kleiner sein als Variablen beider Kindknoten
  - Auf jedem Pfad kommt jede Variable höchstens einmal vor
- Reduktionen:
  - Knoten eindeutig dargestellt
  - Knoten  $k$  mit  $k_h = k_l$  werden gelöscht und Kanten, die auf  $k$  zeigen werden auf  $k_h$  weitergezogen.



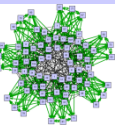
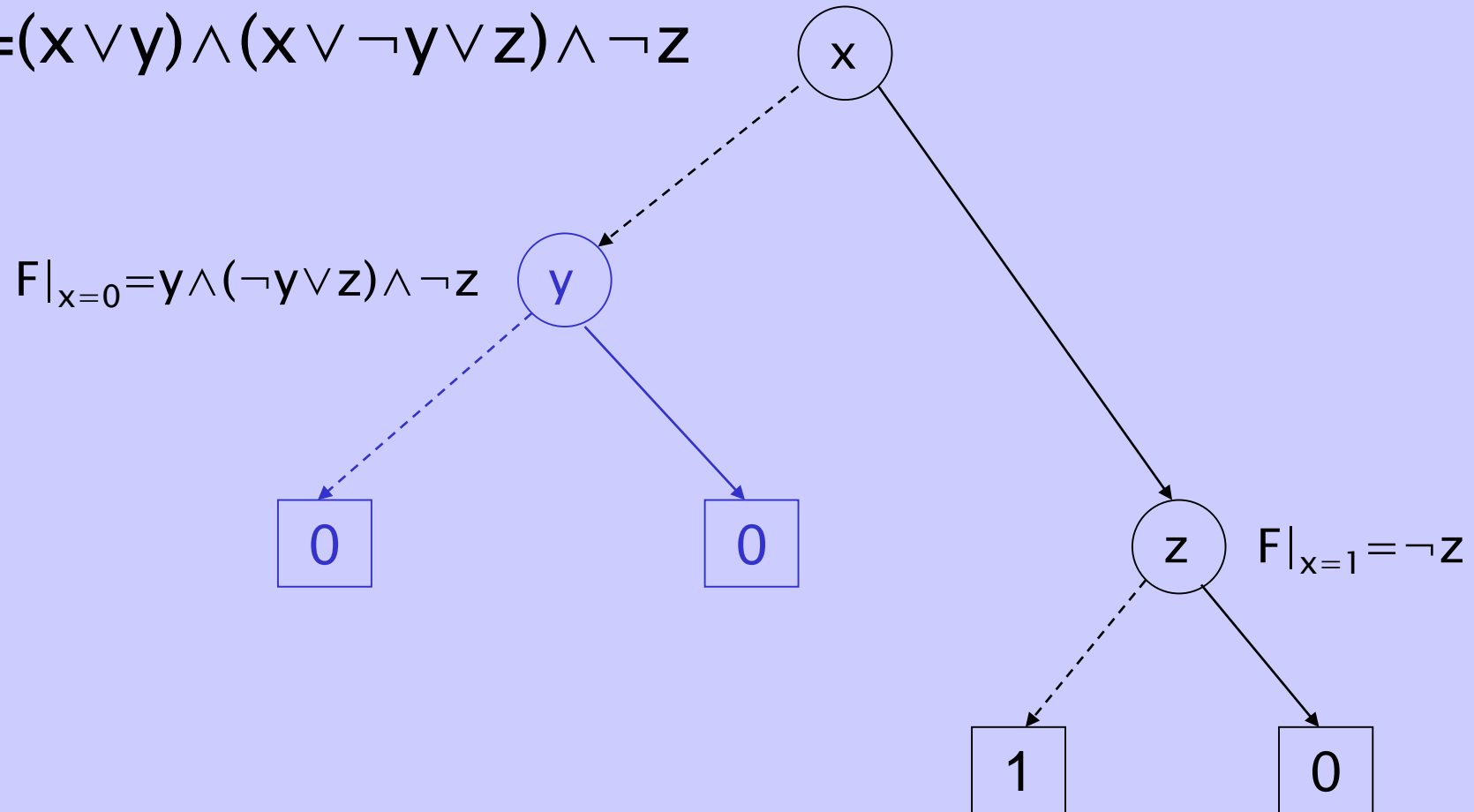
# ROBDDs: Beispiel zur Reduktion eines BDDs

➤  $F = (x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg z$



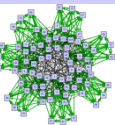
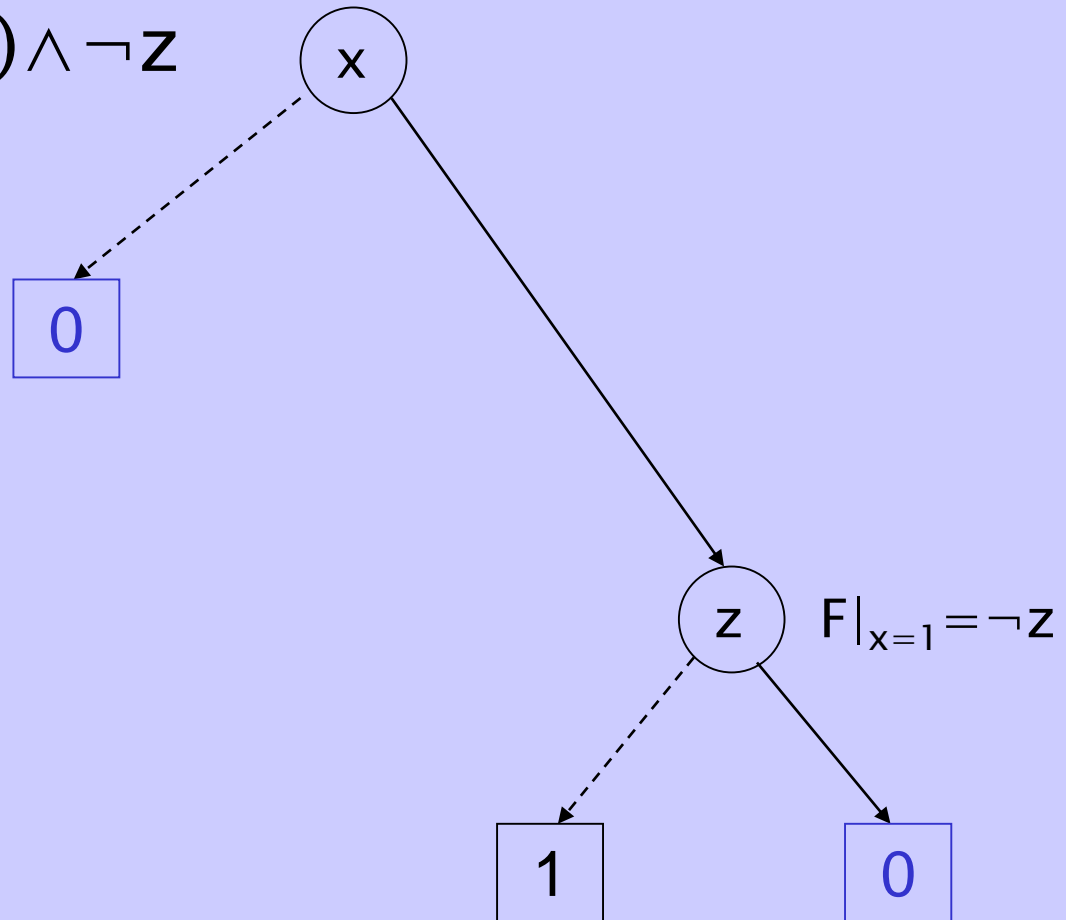
# ROBDDs: Beispiel zur Reduktion eines BDDs

➤  $F = (x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg z$



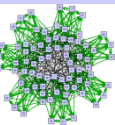
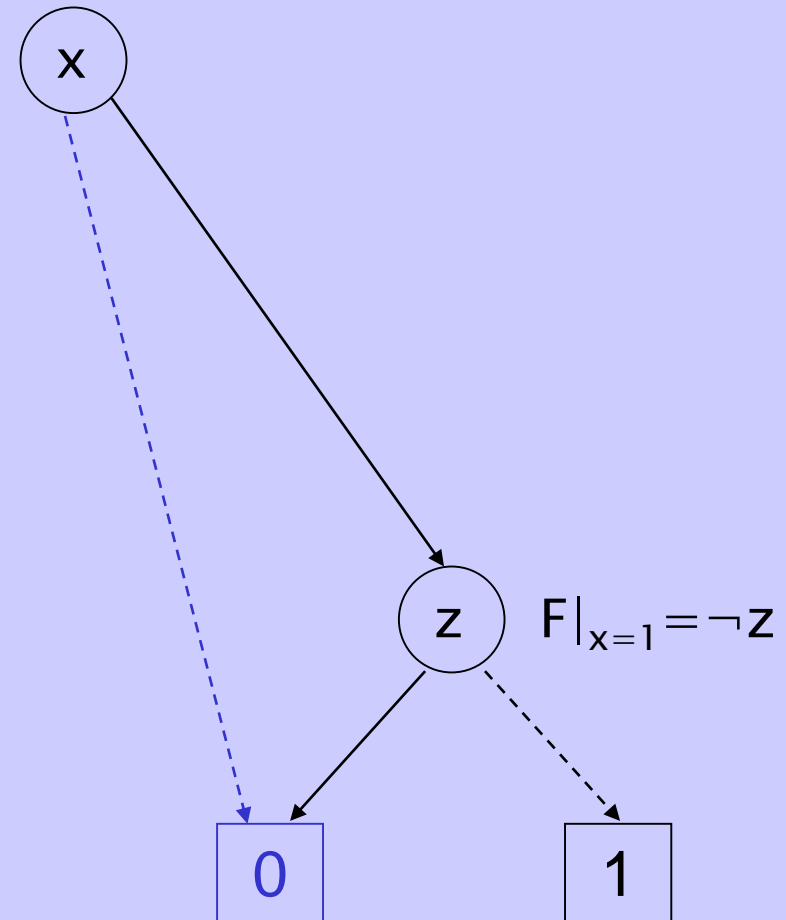
# ROBDDs: Beispiel zur Reduktion eines BDDs

➤  $F = (x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg z$



# ROBDDs: Beispiel zur Reduktion eines BDDs

➤  $F = (x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg z$



# Vom BDD zur Booleschen Formel

---

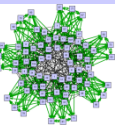
- Wie können wir aus einem BDD die zugehörige Boolesche Formel ablesen?
- Es gibt vier (äquivalente) Interpretationen für einen Knoten im BDD (vgl. Shannon-Expansion):

$$F \equiv (x \wedge F|_{x=1}) \vee (\neg x \wedge F|_{x=0}) \quad // \rightarrow \text{Gewinnung der DNF}$$

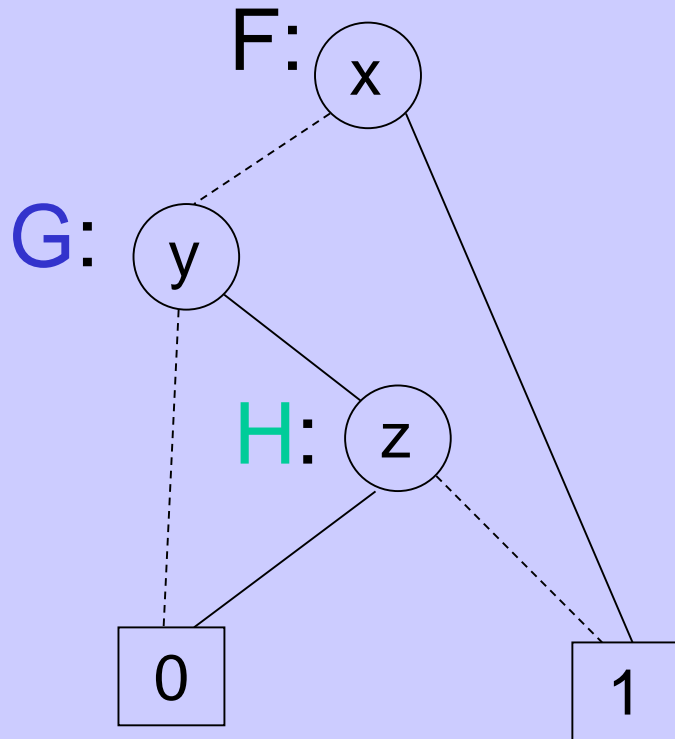
$$\equiv (\neg x \vee F|_{x=1}) \wedge (x \vee F|_{x=0}) \quad // \rightarrow \text{Gewinnung der CNF}$$

$$\equiv (x \Rightarrow F|_{x=1}) \wedge (\neg x \Rightarrow F|_{x=0})$$

$$\equiv \text{if } x \text{ then } F|_{x=1} \text{ else } F|_{x=0} \text{ fi}$$



# Formelgewinnung aus ROBDD



$F = \text{if } x \text{ then } \top \text{ else } G$

$= \text{if } x \text{ then } \top \text{ else (if } y \text{ then } H \text{ else } \perp)$

$= \text{if } x \text{ then } \top \text{ else}$

$(\text{if } y \text{ then (if } z \text{ then } \perp \text{ else } \top) \text{ else } \perp)$

$= \text{if } x \text{ then } \top \text{ else (if } y \text{ then } \neg z \text{ else } \perp)$

$= \text{if } x \text{ then } \top \text{ else } (y \Rightarrow \neg z) \wedge (\neg y \Rightarrow \perp)$

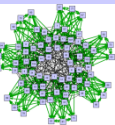
$= \text{if } x \text{ then } \top \text{ else } (y \Rightarrow \neg z) \wedge y$

$= \text{if } x \text{ then } \top \text{ else } \neg z \wedge y$

$= (x \Rightarrow \top) \wedge (\neg x \Rightarrow \neg z \wedge y)$

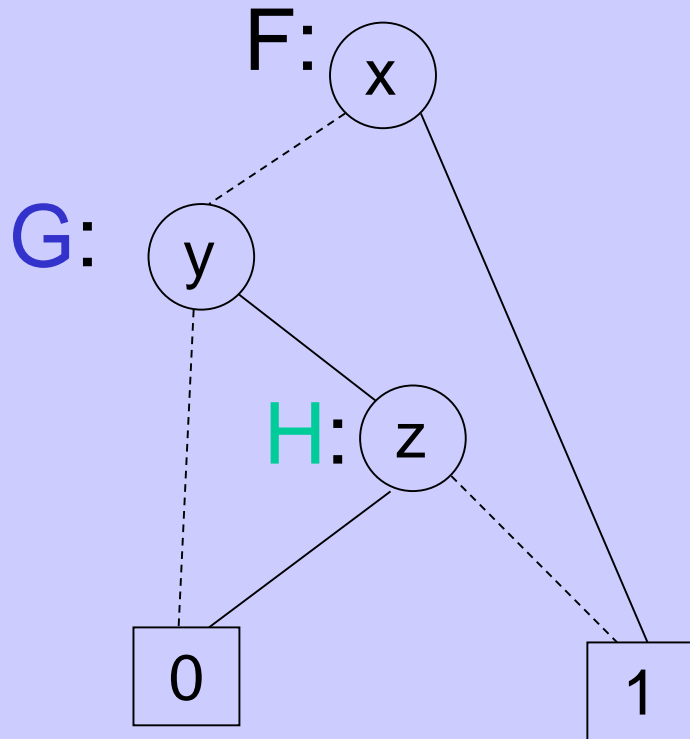
$= x \vee (\neg z \wedge y)$

BDD-Darstellung  
von  $x \vee (y \wedge \neg z)$





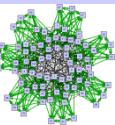
# DNF Gewinnung aus ROBDD



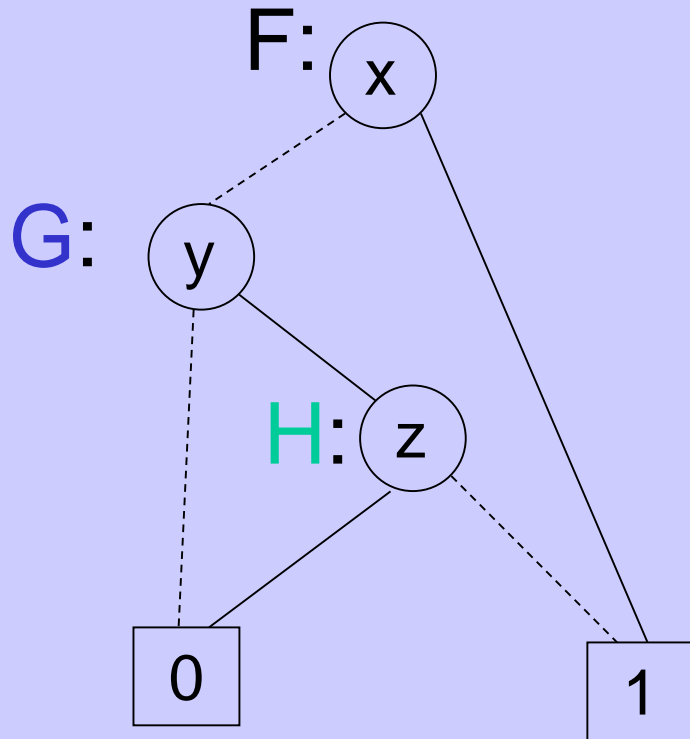
- Wir benutzen  $(x \wedge F|_{x=1}) \vee (\neg x \wedge F|_{x=0})$
- $\text{DNF}(F) = (x \wedge \text{DNF}(F_{hi})) \vee (\neg x \wedge \text{DNF}(F_{lo}))$

$$\begin{aligned}
 F &= (x \wedge T) \vee (\neg x \wedge \text{DNF}(G)) \\
 &= x \vee (\neg x \wedge ((y \wedge \text{DNF}(H)) \vee (\neg y \wedge \perp))) \\
 &= x \vee (\neg x \wedge y \wedge \text{DNF}(H) \vee \neg x \wedge \neg y \wedge \perp) \\
 &= x \vee (\neg x \wedge y \wedge \text{DNF}(H)) \\
 &= x \vee (\neg x \wedge y \wedge ((z \wedge \perp) \vee (\neg z \wedge T))) \\
 &= x \vee (\neg x \wedge y \wedge (z \wedge \perp) \vee \neg x \wedge y \wedge (\neg z \wedge T)) \\
 &= x \vee (\neg x \wedge y \wedge \neg z) \\
 &= (x \vee \neg x) \wedge (x \vee (y \wedge \neg z)) \\
 &= x \vee (\neg z \wedge y)
 \end{aligned}$$

BDD-Darstellung  
von  $x \vee (y \wedge \neg z)$



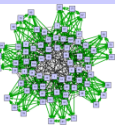
# DNF Gewinnung aus ROBDD (2)



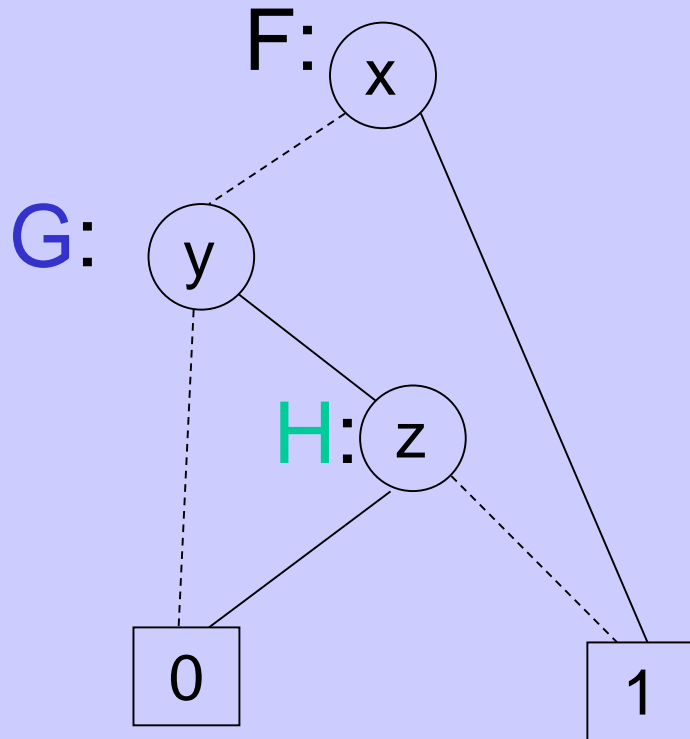
Tiefensuche nach dem 1-Knoten,  
left most – inner most, Backtracking:

$$\begin{aligned} F &= (\neg x \wedge y \wedge \neg z) \vee x \\ &= (x \vee \neg x) \wedge (x \vee (y \wedge \neg z)) \\ &= x \vee (y \wedge \neg z) \end{aligned}$$

BDD-Darstellung  
von  $x \vee (y \wedge \neg z)$



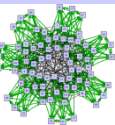
# CNF Gewinnung aus ROBDD



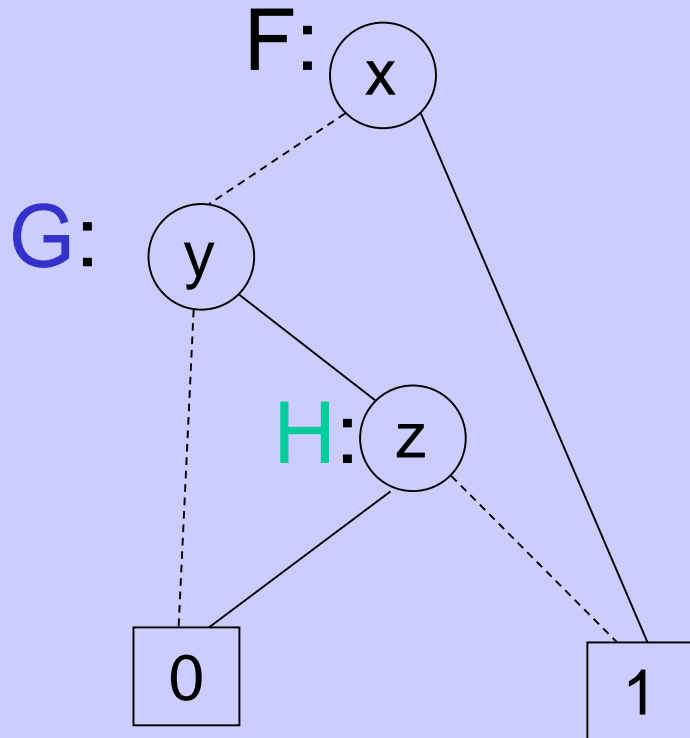
BDD-Darstellung  
von  $x \vee (y \wedge \neg z)$

- Wir benutzen  $(\neg x \vee F|_{x=1}) \wedge (x \vee F|_{x=0})$
- $\text{CNF}(F) = (\neg x \vee \text{CNF}(F_{hi})) \wedge (x \vee \text{CNF}(F_{lo}))$

$$\begin{aligned}
 F &= (\neg x \vee T) \wedge (x \vee \text{CNF}(G)) \\
 &= x \vee ((\neg y \vee \text{CNF}(H)) \wedge (y \vee \perp)) \\
 &= (x \vee \neg y \vee \text{CNF}(H)) \wedge (x \vee (y \vee \perp)) \\
 &= (x \vee \neg y \vee \text{CNF}(H)) \wedge (x \vee y) \\
 &= (x \vee \neg y \vee ((\neg z \vee \perp) \wedge (z \vee T))) \wedge (x \vee y) \\
 &= (x \vee \neg y \vee \neg z \vee \perp) \wedge (x \vee \neg y \vee z \vee T) \wedge (x \vee y) \\
 &= (x \vee \neg y \vee \neg z) \wedge (T) \wedge (x \vee y) \\
 &= (x \vee \neg y \vee \neg z) \wedge (x \vee y) \\
 &= x \vee ((\neg y \vee \neg z) \wedge (y)) \\
 &= x \vee ((\neg y \wedge y) \vee (\neg z \wedge y)) \\
 &= x \vee (\neg z \wedge y)
 \end{aligned}$$



# CNF Gewinnung aus ROBDD (2)



Idee: CNF ist die Konjunktion aus Allem, was nicht passieren darf.

Also: Simultaner Ausschluss der Wege zur 0

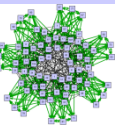
$$F = \neg(\neg x \wedge \neg y) \wedge \neg(\neg x \wedge y \wedge z)$$

$$\equiv (x \vee y) \wedge (x \vee \neg y \vee \neg z)$$

$$\equiv x \vee (y \wedge (\neg y \vee \neg z))$$

$$\equiv x \vee (y \wedge \neg z)$$

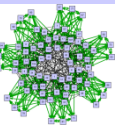
BDD-Darstellung  
von  $x \vee (y \wedge \neg z)$



# ROBDDs: Größe

---

- Die Größe des ROBDDs kann stark von der gewählten Variablenordnung abhängen:
  - Bryant: Das ROBDD für die Formel  $(p_1 \wedge p_2) \vee \dots \vee (p_{2n-1} \wedge p_{2n})$  hat  $2n+2$  Knoten unter der Ordnung  $p_1, p_2, \dots, p_{2n}$  und  $2^{n+1}$  Knoten unter der Ordnung  $p_1, p_{n+1}, p_2, p_{n+2}, \dots, p_n, p_{2n}$ .
  - Bryant: Es gibt  $F$  mit  $n$  Variablen, so dass ROBDD zu  $F$  für jede Ordnung mindestens  $2^{cn}$  Knoten hat ( $c > 0$ ).



# Generierung von ROBDDs

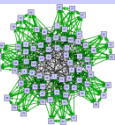
---

## ➤ „Top-Down“-Ansatz

- Zerlegung der Eingabeformel  $F$  unter gleichzeitiger Generierung des BDDs, d.h.
  - Selektion der kleinsten in  $F$  vorkommenden Variablen
  - Anlegen eines Knotens und Berechnung von  $F_l$  und  $F_h$
  - Rekursive BDD-Transformation von  $F_l$  und  $F_h$
  - Einbau der Kind-BDDs in den Knoten und Reduktion

## ➤ „Bottom-Up“-Ansatz

- BDDs für atomare Formeln (Variablen,  $\perp$ ,  $\top$ )
- Operationen zur Generierung komplexer BDDs aus einfacheren (z.B. BDD-Disjunktion, -Konjunktion, -Negation)



# Effiziente Konstruktion von ROBDDs

## ➤ Aufbau eines ODT (ordered decision tree)

Sei  $a$  ein boolescher Ausdruck in  $n$  Variablen.

$\text{Build}(a,n) \equiv \text{Build}'(a,1,n)$

$\text{Build}'(a,i,n) \equiv$

if  $(i > n)$  then // all variables are substituted,  $a$  evaluates to true or false

if  $a$  then return 1 else return 0 fi

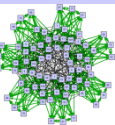
else // build BDDs for  $\text{low}(x_i)$  and  $\text{hi}(x_i)$  and build node  $x_i$

$\text{MkNode}(i, \text{Build}'(a[x_i/0], i+1, n), \text{Build}'(a[x_i/1], i+1, n))$

fi

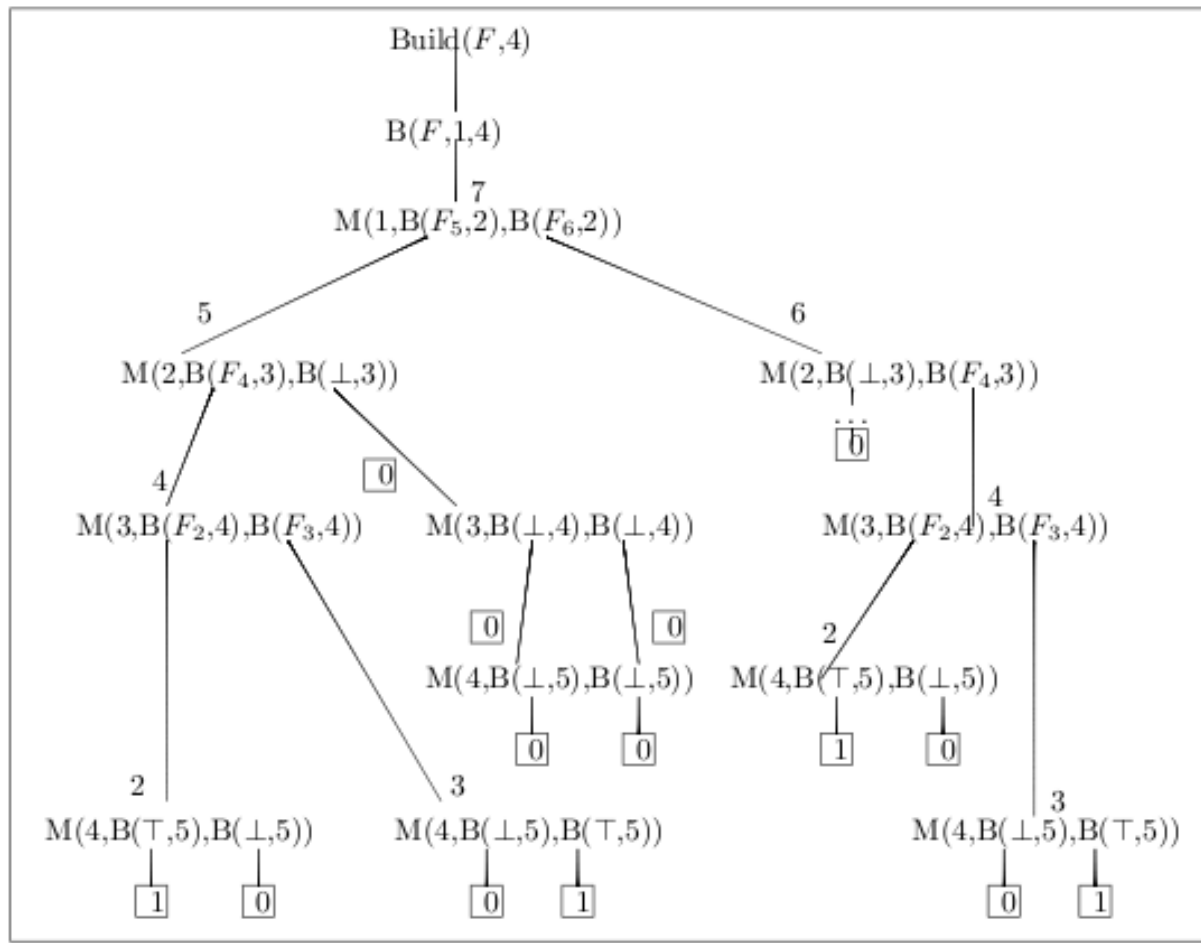
## ➤ Bei naiver Implementierung von $\text{MkNode}$ wird ein Baum aufgebaut mit $2^n$ Knoten

- Gleiche Knoten werden mehrfach erzeugt



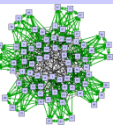
# Beispiel zum Aufbau eines ODT

➤  $F = (\neg x \wedge \neg y \wedge (z \leftrightarrow u)) \vee (x \wedge y \wedge (z \leftrightarrow u))$ ; Ordnung:  $x < y < z < u$



- $F_5 = F[x=0] = (\neg y \wedge (z \leftrightarrow u))$
- $F_6 = F[x=1] = (y \wedge (z \leftrightarrow u))$
- $F_4 = F_5[y=0] = (z \leftrightarrow u)$
- $F_5[y=1] = \perp$
- $F_2 = F_4[z=0] = (\perp \leftrightarrow u)$
- $F_2[u=0] = \top$
- $F_2[u=1] = \perp$
- $F_3 = F_4[z=1] = (\top \leftrightarrow u)$
- $F_3[u=0] = \perp$
- $F_3[u=1] = \top$
- $F_6[y=0] = \perp$
- $F_6[y=1] = (z \leftrightarrow u) = F_4$

B steht für Build' und M für MkNode.  
z.T. ist der 3. Parameter von B weggelassen





# Konstruktion von ROBDDs (2)

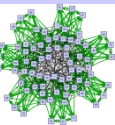
## ➤ Effizienter Aufbau von ROBDDs

- Kein unnötiger rek. Aufruf  $\text{Build}'(a, i, n)$  falls  $a = \top$  oder  $a = \perp$
- Jeder Knoten darf nur einmal konstruiert werden
- Tabellenrepräsentation, um existierende Knoten schnell zu finden.

## ➤ Operationen auf Tabelle T:

- $\text{init}(T)$ : füge 0 und 1 in leeres T ein
- $u \leftarrow \text{add}(i, l, h)$ : neuer Knoten
- $\text{var}(u)$ ,  $\text{low}(u)$ ,  $\text{high}(u)$ : accessors

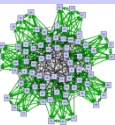
Node #	Var #	l	h
0	n+1		
1	n+1		
2	4	1	0
3	4	0	1
4	3	2	3
5	2	4	0
6	2	0	4
7	1	5	6



# Konstruktion von ROBDDs (3)

---

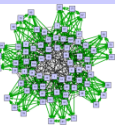
- Zum Auffinden existierender Knoten inverse Tabelle H:
  - $\text{init}(H)$ : leeres H anlegen
  - $b \leftarrow \text{member}(i,l,h)$ : ist  $(i,l,h)$  in H vorhanden?
  - $u \leftarrow \text{lookup}(i,l,h)$ : finde  $H(i,l,h)$
  - $\text{insert}(i,l,h,u)$ : repräsentiere Abb.  $(i,l,h) \equiv u$  durch Eintrag in H
- H muss effizient als Hashmap realisiert werden
  - sonst könnte man auch in Tabelle T suchen ...



# Intelligentes MakeNode für ROBDDs

---

```
MkNode[T,H](i,l,h) :  $\equiv$   
  if (l=h) then return h  
  else if member[H](i,l,h) then return lookup[H](i,l,h)  
    else u = add[T](i,l,h);  
      insert[H](i,l,h,u);  
      return u;  
    fi  
  fi
```



# Beispiel zum Aufbau eines ROBDDs

➤  $F = (\neg x \wedge \neg y \wedge (z \leftrightarrow u)) \vee (x \wedge y \wedge (z \leftrightarrow u))$

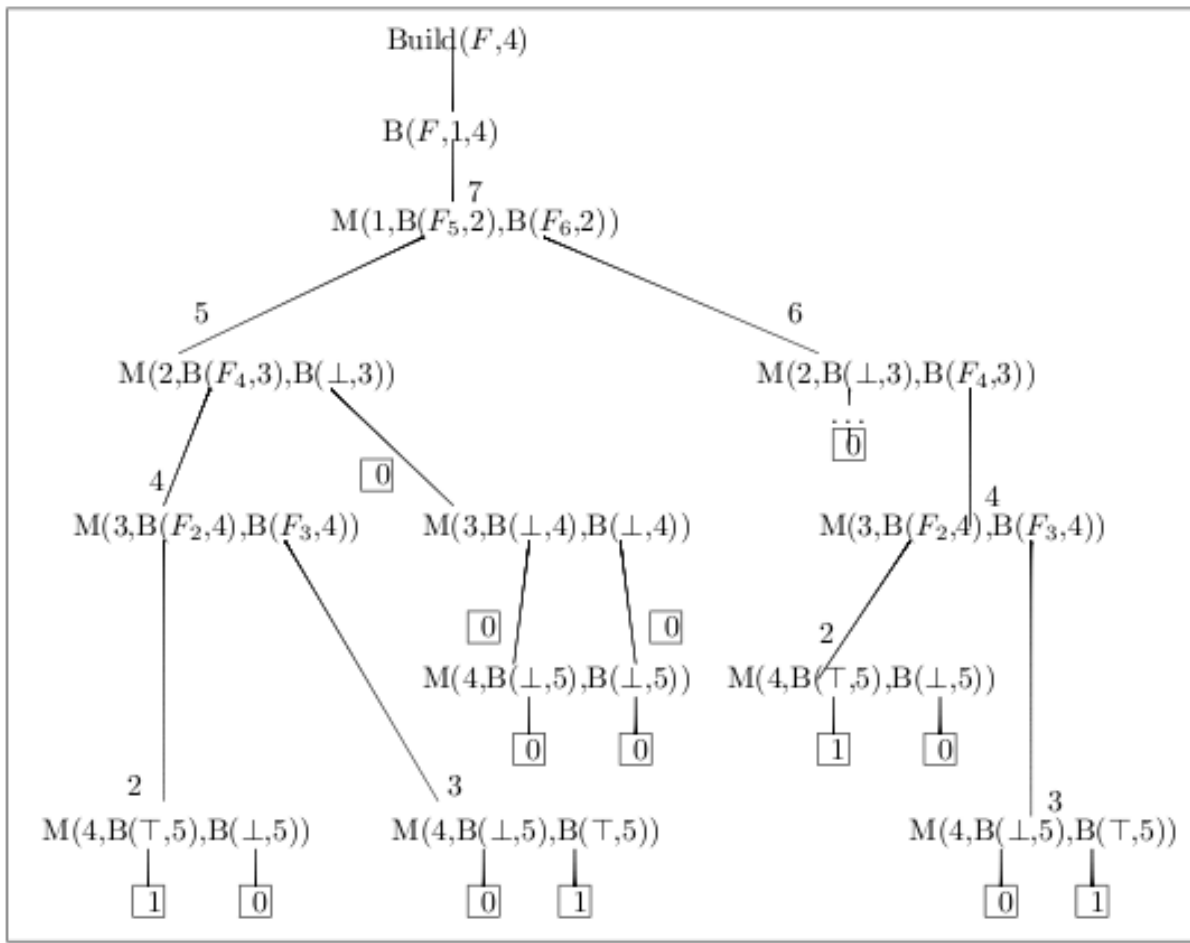
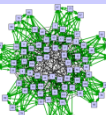


Tabelle T

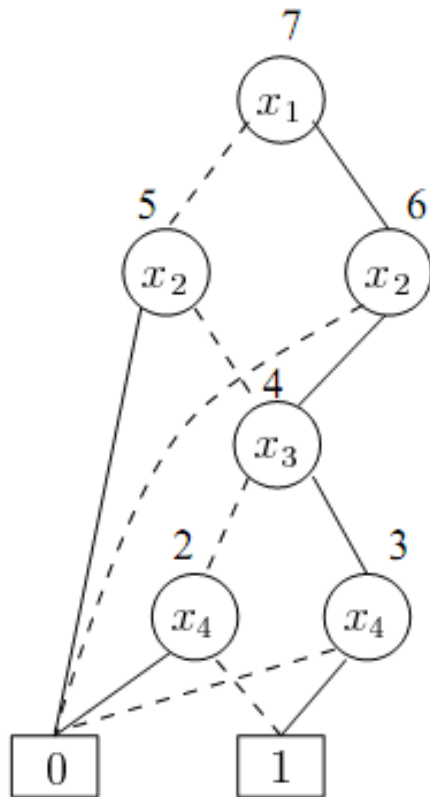
Node #	Var #	l	h
0	n+1		
1	n+1		
2	4	1	0
3	4	0	1
4	3	2	3
5	2	4	0
6	2	0	4
7	1	5	6

B steht für Build' und M für MkNode  
z.T. ist der 3. Parameter von B weggelassen.



# Beispiel zum Aufbau eines ROBDDs

➤  $F = (\neg x \wedge \neg y \wedge (z \leftrightarrow u)) \vee (x \wedge y \wedge (z \leftrightarrow u))$



$T : u \rightarrow (i, l, h)$

$u$	$var$	$low$	$high$
0	5		
1	5		
2	4	1	0
3	4	0	1
4	3	2	3
5	2	4	0
6	2	0	4
7	1	5	6



# Boolesche Operationen auf ROBDDs

---

## ➤ Apply(op, u1, u2)

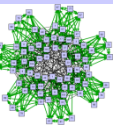
- Wende boolesche Operation op auf BDD u1 und BDD u2 an
- $\text{Apply}(\text{op}, \text{BDD}(t1), \text{BDD}(t2)) = \text{BDD}(t1 \text{ op } t2)$

## ➤ Grundlagen:

- $(\text{if } x \text{ then } t1 \text{ else } t2 \text{ fi}) \text{ op } (\text{if } x \text{ then } t'1 \text{ else } t'2 \text{ fi})$   
 $\equiv \text{if } x \text{ then } (t1 \text{ op } t'1) \text{ else } (t2 \text{ op } t'2) \text{ fi}$
- $(\text{if } x \text{ then } t1 \text{ else } t2 \text{ fi}) \text{ op } t$   
 $\equiv \text{if } x \text{ then } t1 \text{ op } t \text{ else } t2 \text{ op } t \text{ fi}$

## ➤ Praxis:

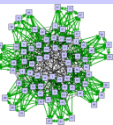
- Verwende (Hash-)Tabelle G zur Speicherung von Zwischenergebnissen (dyn. Programmieren)
- Nicht alle Variablen sind in beiden BDDs gleichzeitig vorhanden



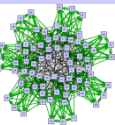
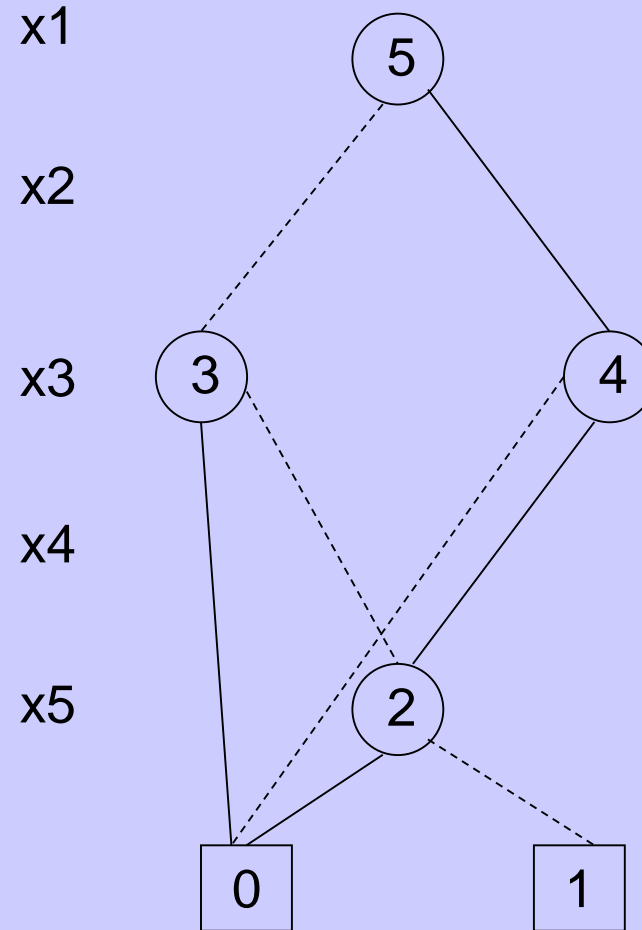
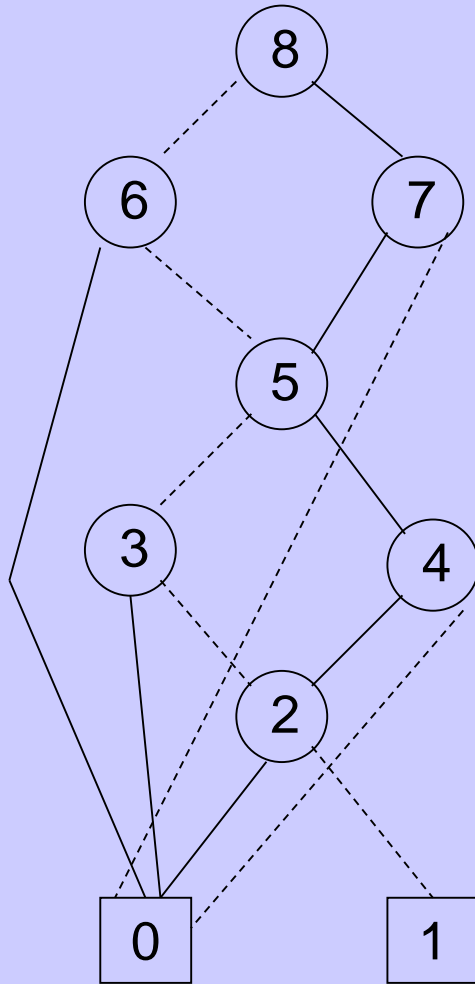
# Apply für ROBDDs

$\text{Apply}[T, H](\text{op}, u_1, u_2) \equiv \text{init}(G); \text{App}[T, H](\text{op}, u_1, u_2)$

```
App[T, H](op, u1, u2)  $\equiv$ 
  if G(u1, u2)  $\neq$  empty then return G(u1, u2)
  else if u1  $\in$  {0,1} and u2  $\in$  {0,1} then u  $\leftarrow$  op(u1, u2)
    else if var(u1) = var(u2) then
      u  $\leftarrow$  MkNode(var(u1), App(op, lo(u1), lo(u2)),
        App(op, hi(u1), hi(u2)))
    else if var(u1) < var(u2) then
      u  $\leftarrow$  MkNode(var(u1), App(op, lo(u1), u2),
        App(op, hi(u1), u2))
    else u  $\leftarrow$  MkNode(var(u2), App(op, u1, lo(u2)),
      App(op, u1, hi(u2)))
    fi
  fi
fi
G(u1, u2)  $\leftarrow$  u
return(u)
```

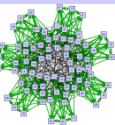
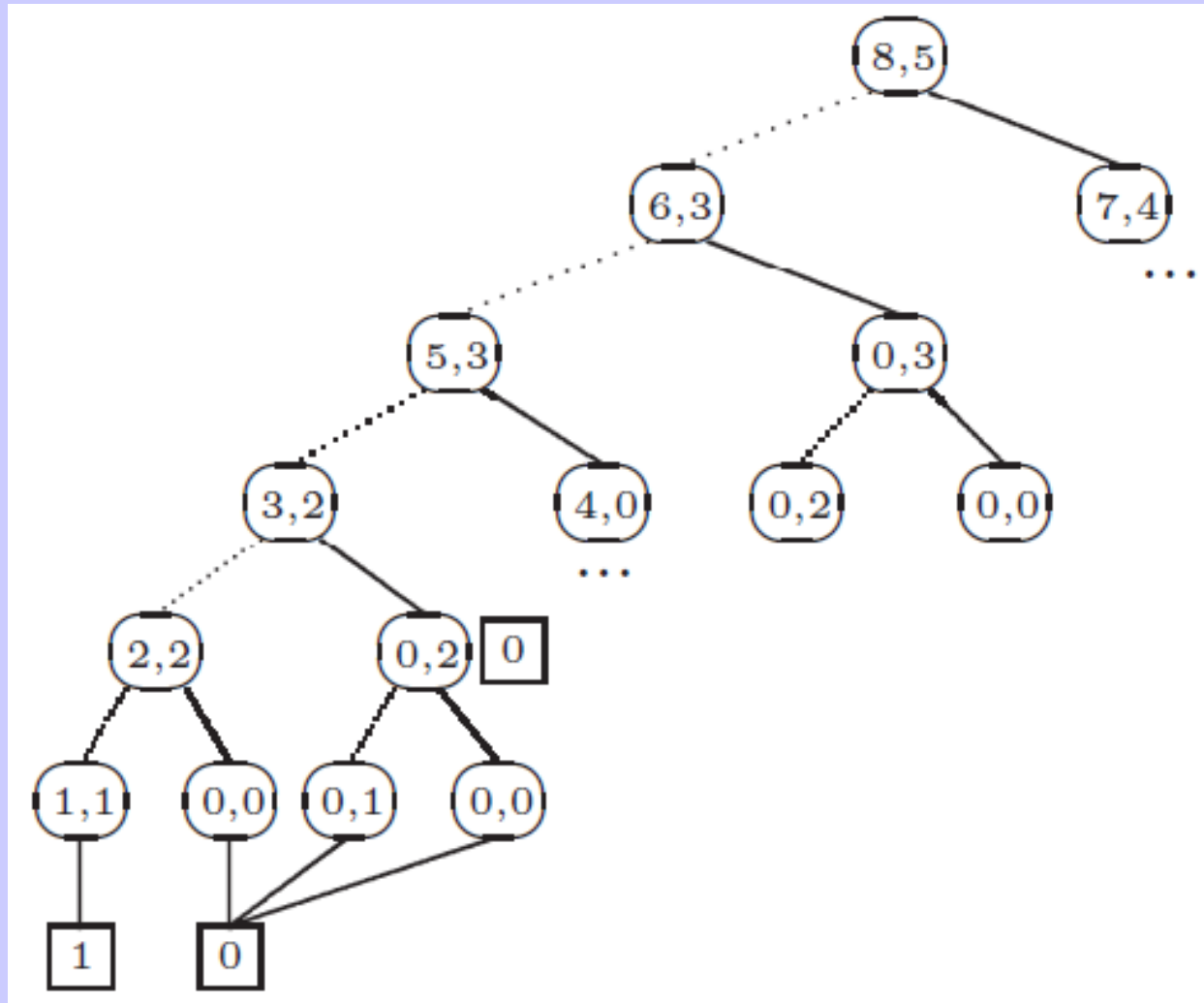


# Beispiel zu Apply: $\wedge$ -Verknüpfung





# Baum aller Aufrufe im obigen Beispiel

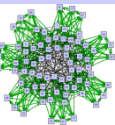


# Restriktion auf BDDs

---

- Algorithmus *Restrict*( $u, j, b$ ), erzeugt zu einem OBDD  $u$  einen OBDD für  $u|_{j=b}$ .

```
Restrict[T,H](u, j, b) :=  
if var(u) > j then return u // j not in u  
else if var(u) < j then // build new OBDD from restricted parts  
    return MkNode(var(u), Restrict(lo(u),j,b), Restrict(hi(u), j,b))  
else if b=0 then return lo(u) // var(u) = j, j:=0  
else if b=1 then return hi(u) // var(u) = j, j:= 1  
fi fi fi fi
```



# Quantifizierung

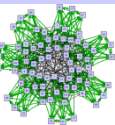
---

➤ existentielle:  $\exists x F := F|_{x=0} \vee F|_{x=1}$

$\text{OBDD}(\exists x F) = \text{Apply}(\text{or}, \text{Restrict}(F, x, 0), \text{Restrict}(F, x, 1))$

➤ universelle:  $\forall x F := F|_{x=0} \wedge F|_{x=1}$

$\text{OBDD}(\forall x F) = \text{Apply}(\text{and}, \text{Restrict}(F, x, 0), \text{Restrict}(F, x, 1))$



# Anzahl der erfüllenden Belegungen

---

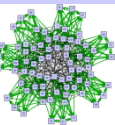
SatCount(u) =

$$2^{\text{var}(\text{low}(u)) - \text{var}(u) - 1} * \text{SatCount}(\text{low}(u)) +$$

$$2^{\text{var}(\text{high}(u)) - \text{var}(u) - 1} * \text{SatCount}(\text{high}(u))$$

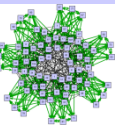
- $\text{var}(\text{low}(u)) - \text{var}(u) - 1$  zählt die übersprungenen Variablen beim Übergang von  $u$  zu  $\text{low}(u)$
- Ohne übersprungene Variablen ergibt sich:

$$\text{SatCount}(u) = \text{SatCount}(\text{low}(u)) + \text{SatCount}(\text{high}(u))$$



# Konsistenztest auf ROBDDs

- Sei  $F$  eine boolesche Formel.  $G$  ist konsistent mit  $F$ , falls  $G \wedge F$  erfüllbar ist.
- Falls  $F$  und  $G$  als BDD  $u_1$ ,  $u_2$  vorliegen, haben wir
  - $\text{Consistent}(F, G) = \text{SAT}(u_1 \wedge u_2) = \text{SAT}(\text{APPLY}(\text{and}, u_1, u_2))$ .
  - Der BDD  $\text{APPLY}(\text{and}, u_1, u_2)$  wird aber nicht wirklich gebraucht
- $\text{AndSat}(u_1, u_2)$ 
  - $\text{SAT}((\text{if } x \text{ then } t_1 \text{ else } t_2 \text{ fi}) \wedge (\text{if } x \text{ then } t'_1 \text{ else } t'_2 \text{ fi}))$   
=  $\text{AndSat}((\text{if } x \text{ then } t_1 \text{ else } t_2 \text{ fi}), (\text{if } x \text{ then } t'_1 \text{ else } t'_2 \text{ fi}))$   
=  $\text{AndSat}(t_1, t'_1) \text{ OR } \text{AndSat}(t_2, t'_2)$
  - $\text{SAT}((\text{if } x \text{ then } t_1 \text{ else } t_2 \text{ fi}) \wedge t)$   
=  $\text{AndSat}((\text{if } x \text{ then } t_1 \text{ else } t_2 \text{ fi}), t)$   
=  $\text{AndSat}(t_1, t) \text{ OR } \text{AndSat}(t_2, t)$



# Konsistenztest für ROBDDs

init(G);

AndSAT[G](u1, u2)  $\equiv$

if  $G(u1, u2) \neq \text{empty}$  then return  $G(u1, u2)$  else

if  $u1 \in \{0,1\}$  and  $u2 \in \{0,1\}$  then return  $(u1 \wedge u2)$

else if  $\text{var}(u1) = \text{var}(u2)$  then

$b \leftarrow (\text{AndSAT}(\text{lo}(u1), \text{lo}(u2)) \vee \text{AndSAT}(\text{hi}(u1), \text{hi}(u2)))$

else if  $\text{var}(u1) < \text{var}(u2)$  then

$b \leftarrow (\text{AndSAT}(\text{lo}(u1), u2) \vee \text{AndSAT}(\text{hi}(u1), u2))$

else  $b \leftarrow (\text{AndSAT}(u1, \text{lo}(u2)) \vee \text{AndSAT}(u1, \text{hi}(u2)))$

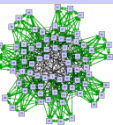
fi

fi

$G(u1, u2) \leftarrow b;$

return b;

fi fi



# Bemerkungen zu AndSAT

---

- Der Test lässt sich wie folgt optimieren  
if  $u1 = 0$  or  $u2 = 0$  then return false  
else if  $u1 = 1$  or  $u2 = 1$  then return true  
else ...
- Der Test lässt sich auf beliebige Mengen von BDDs verallgemeinern
- Der Test lässt sich verallgemeinern auf den Fall, dass  $u2$  nicht als BDD vorliegt, aber die Variablen von  $u2$  eine Teilmenge der Variablen von  $u1$  sind  
(Ersetze  $\text{AndSAT}(\text{lo}(u1), \text{lo}(u2)) \vee \text{AndSAT}(\text{hi}(u1), \text{hi}(u2))$   
durch  $\text{AndSAT}(\text{lo}(u1), u2|_{\text{var}(u1)=0}) \vee \text{AndSAT}(\text{hi}(u1), u2|_{\text{var}(u1)=1})$ )
- Der Test lässt sich so abändern, dass eine DNF von  $u1 \wedge u2$  erzeugt wird (gebe statt „true“ jeweils die Variablenbelegung aus).

