

Computergestützte Verifikation

23.5.03

3.7 Tools

Spin

Eingabesprache: Promela – Prozesse + guarded commands

Spezifikation: LTL oder direkt als Büchi-Automat

Reduktionstechniken: Partial Order Reduction

Sonstige Tricks: Bithashing, Hash Compaction

<http://netlib.bell-labs.com/netlib/spin/whatispin.html>

Murφ

Eingabesprache: Guarded Commands

Spezifikation: Einfache Erreichbarkeitsqueries

Reduktionstechniken: Symmetrie (scalar sets)

sonstige Tricks: parallele Version

<http://verify.stanford.edu/dill/murphi.html>

LOLA

Eingabesprache: Petrinetze

Spezifikation: CTL, viele kleine Eigenschaftsklassen

Reduktionstechniken: stubborn sets, Symmetrien
(Automorphismen)

sonstige Tricks: Petrinetzspezifische Tricks, parallele Version,
Bithashing

<http://www.informatik.hu-berlin.de/~kschmidt/lola.html>

(1. Systeme 2. Spezifikationen 3. explizites Model Checking)

4. Symbolisches Model Checking

4.1 CTL Model Checking mit Binary Decision Diagrams

(4.2 SAT-basiertes Model Checking)

(4.3 Tools)

Binary Decision Diagrams (BDD)

Ausgangspunkt: Boolesche Funktionen

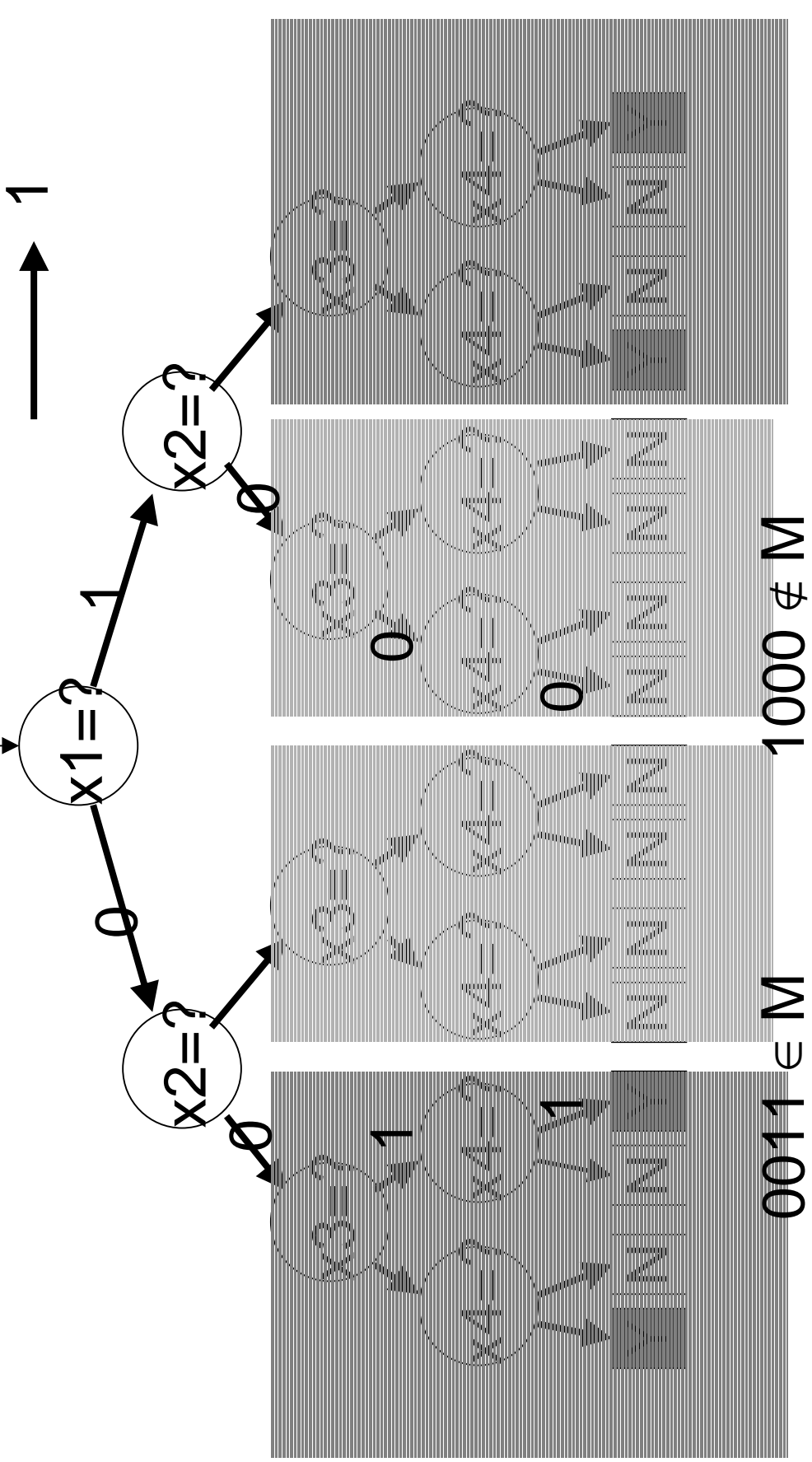
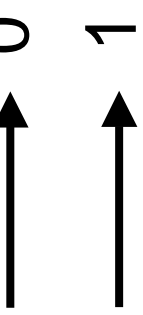
= Mengen von Bit-Vektoren:

$$M \leftrightarrow f_M \quad f_M(x_1, \dots, x_n) = W \text{ gdw. } [x_1, \dots, x_n] \in M$$

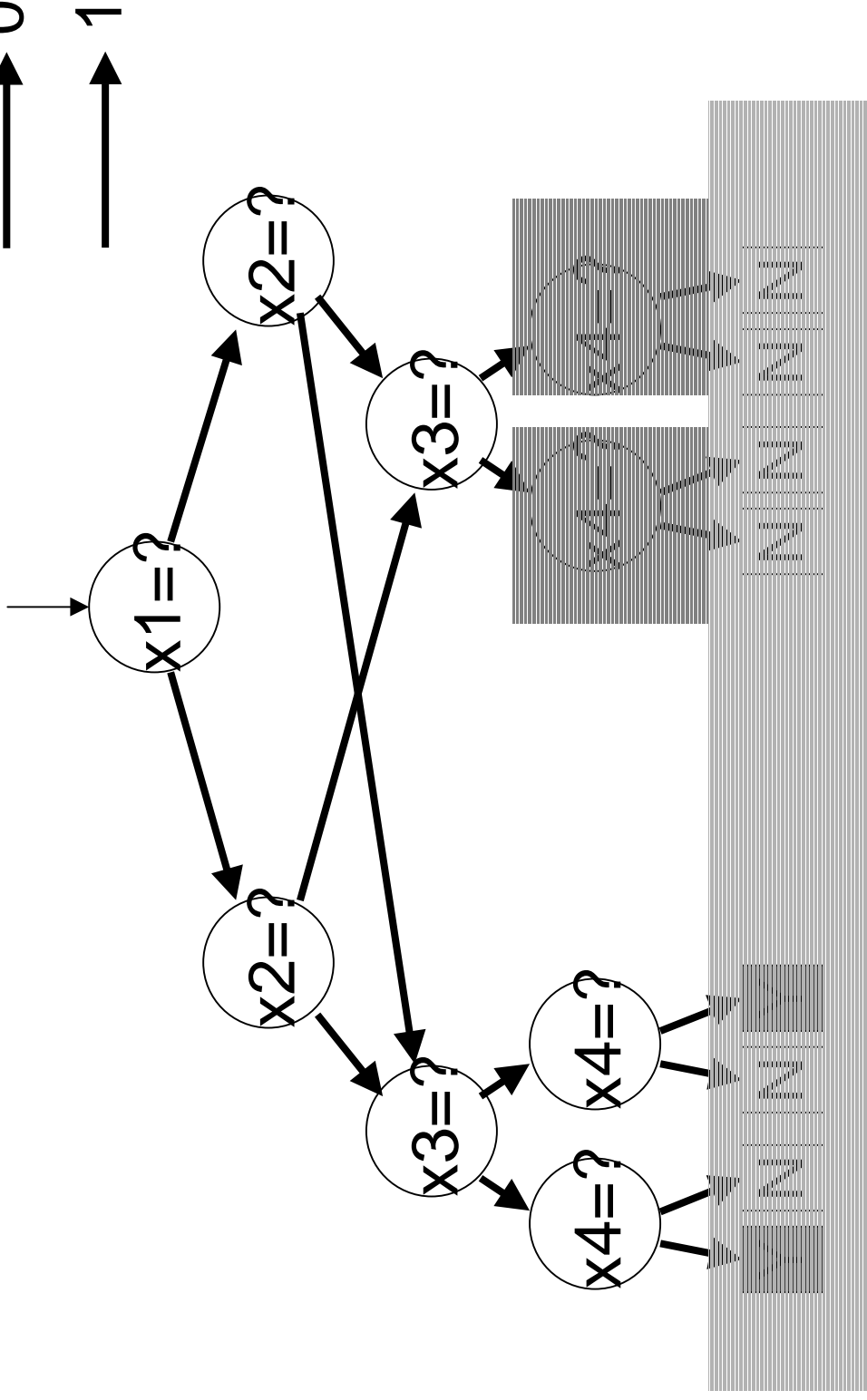
Inhalt:

- 4.1.1 Die Datenstruktur BDD
- 4.1.2 Operationen auf BDD
- 4.1.3 CTL Model Checking mit BDD
- 4.1.4 Fairness

Binary Decision Tree

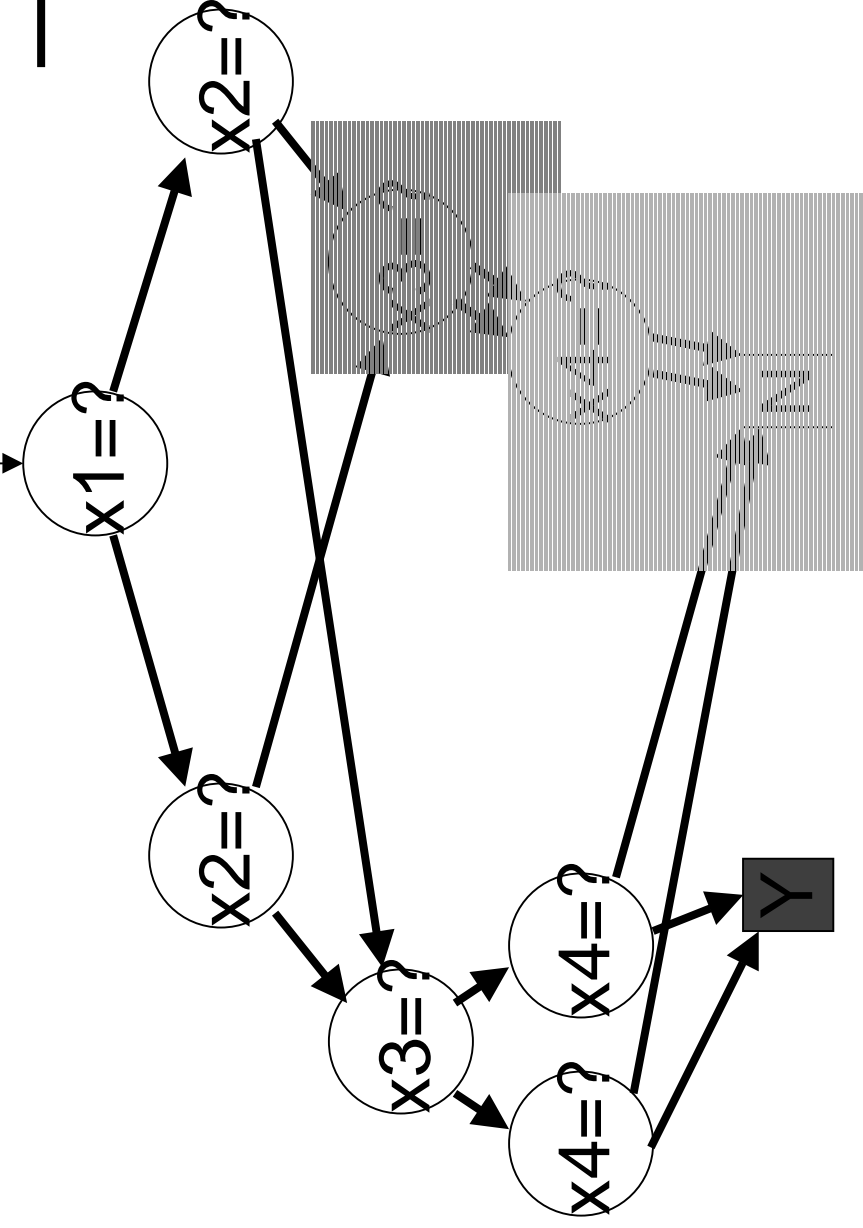


Reduktion des Baums \rightarrow_0



Reduktion des Baums $\rightarrow 0$

$\rightarrow 1$

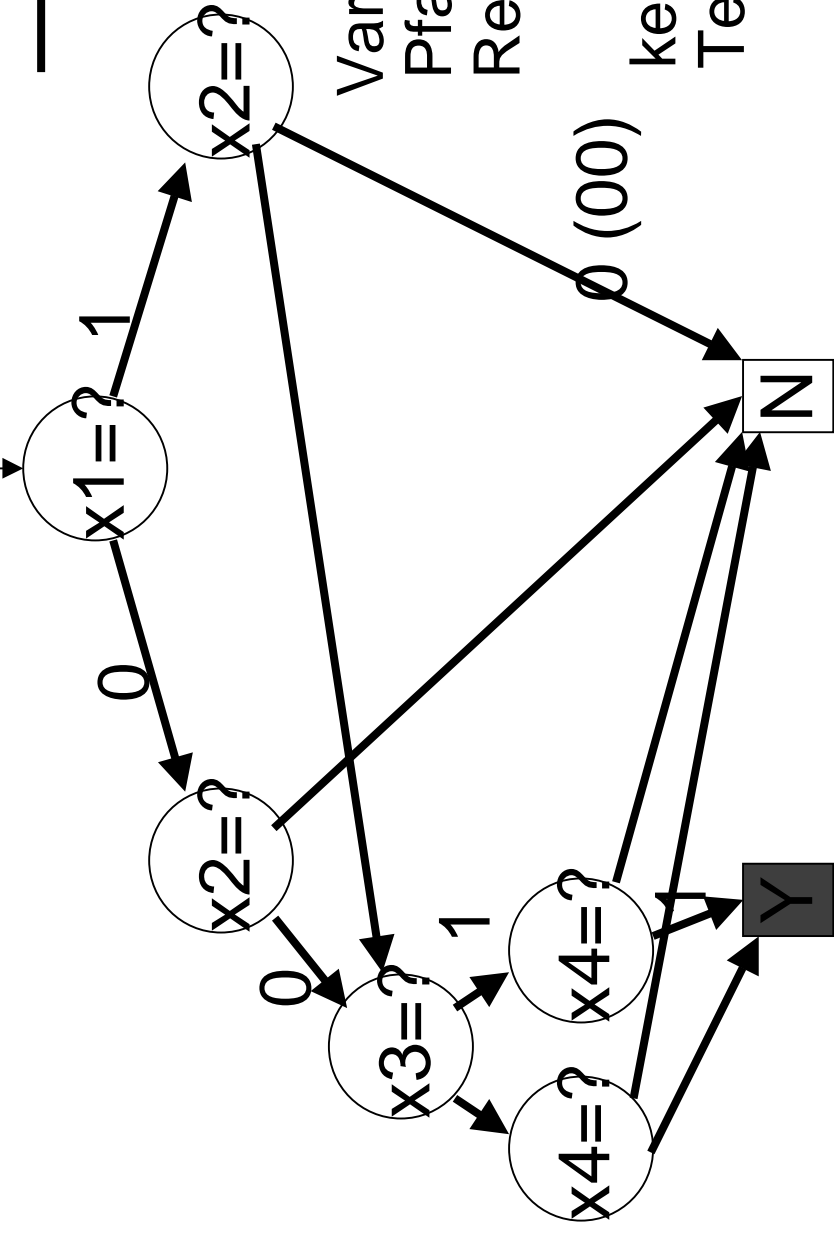


Ordered Binary Decision Diagram

Diagram

→ 0

→ 1



Variablen auf jedem Pfad in gleicher Reihenfolge

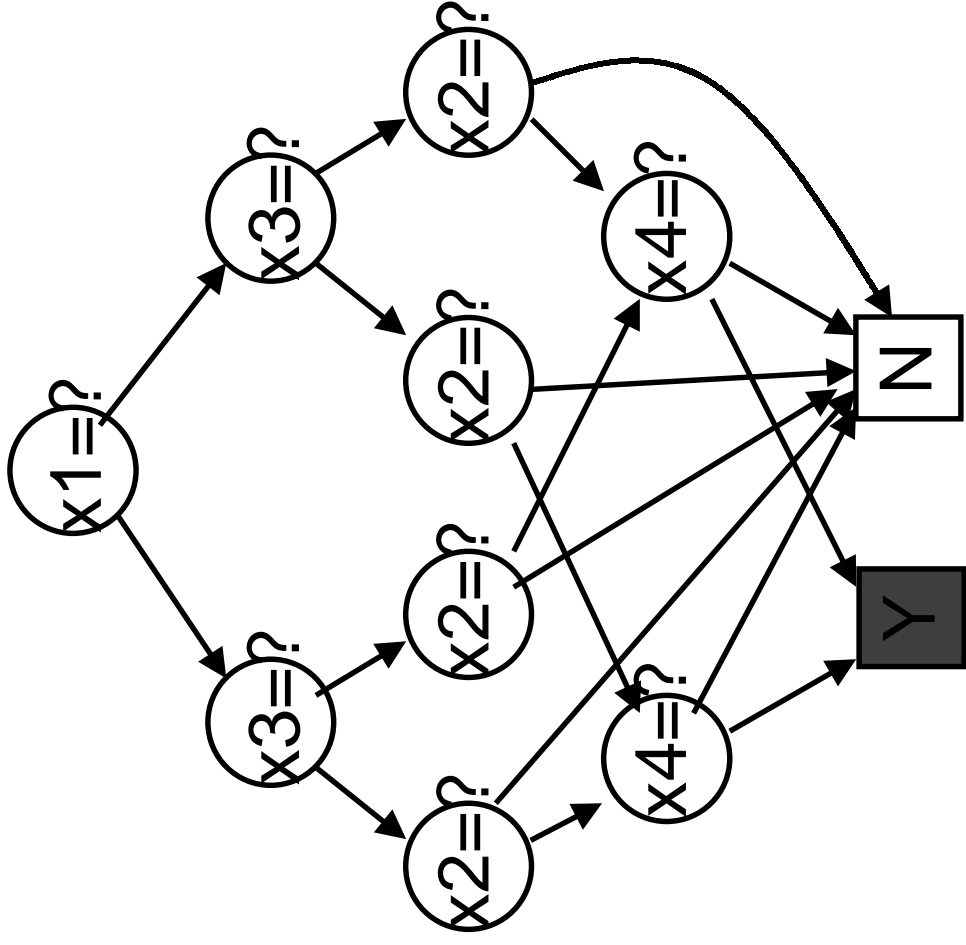
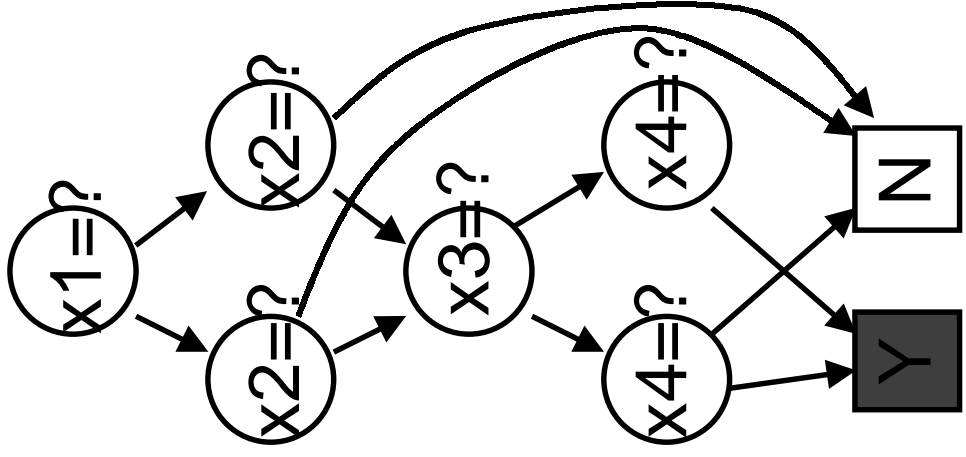
keine äquivalenten Teilbäume

Keine redundanten Knoten

0011 ∈ M

1000 ∉ M

Verschiedene Variablenordnungen



Abhängigkeit von

Variablenordnung

Manche Funktionen haben kleine (polynomiell große) BDD, unabhängig von der Variablenordnung

Manche Funktionen haben (exponentiell) große BDD, unabhängig von der Variablenordnung

Für viele Funktionen kann die Größe des BDD zwischen poly und exp schwanken, abhängig von Variablenordnung

Optimale Variablenordnungen sind nicht effizient berechenbar

Viele, viele Heuristiken

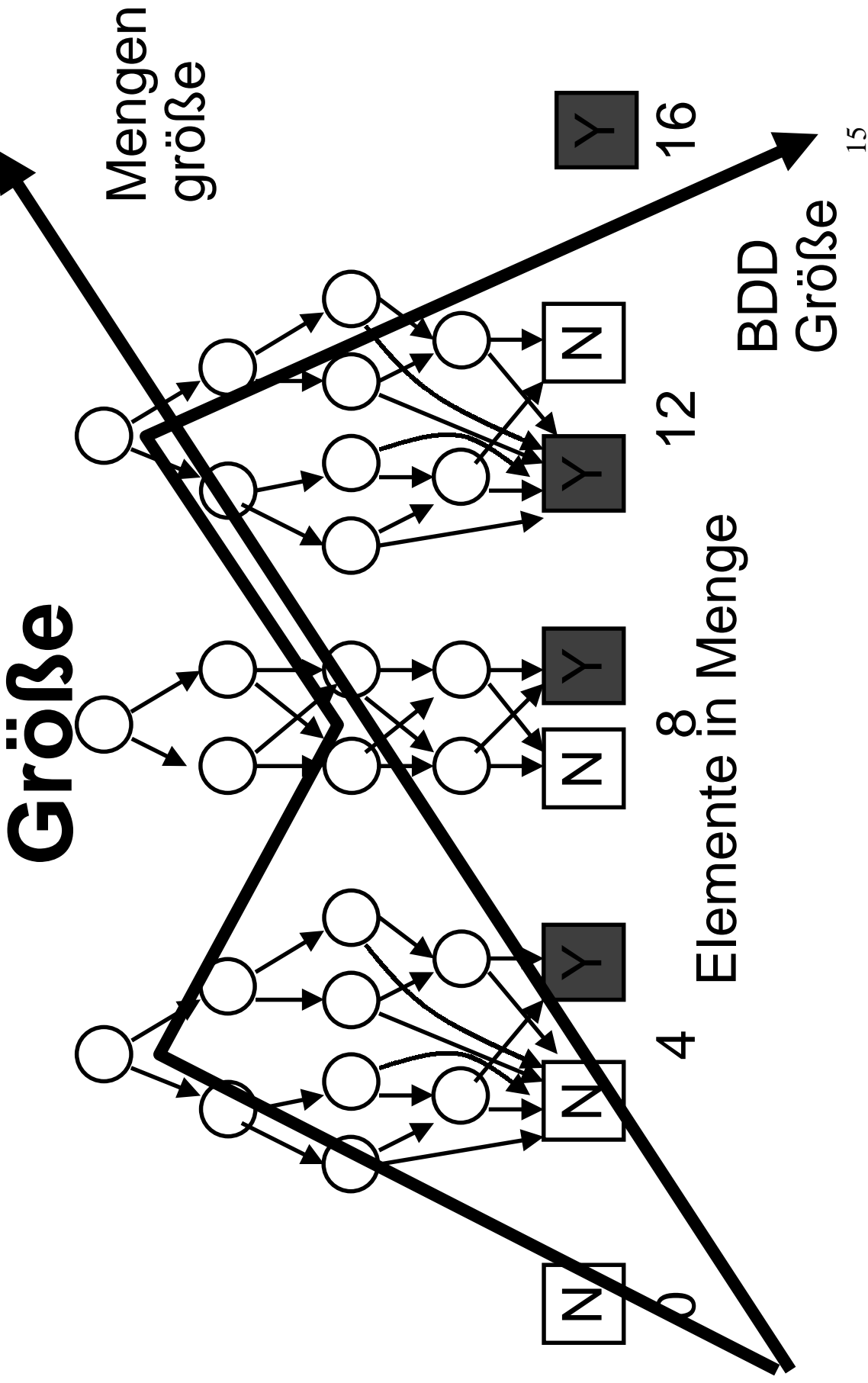
Reordering (statisch oder dynamisch) hilft bei Speicherproblemen

Normalform

Bei gegebener Variablenordnung ist das entstehende BDD **eindeutig** bestimmt!

→ Mengengleichheit, Emptiness können effizient entschieden werden

Mengengröße vs. BDD-Größe



4.1.2 Operationen auf BDD

Eingabe: 1-2 (reduzierte!) BDD

Ausgabe: neues (auch reduziertes!) BDD

einfaches RESTRICT:

f n-stellig b in {0,1}
RESTRICT(f,b) (n-1)-stellig
f'(x₂,...,x_n) := f(b,x₂,...,x_n)

Implementation: root := root.child[b]

Operationen auf BDD

APPLY(op, f1, f2)

f1, f2 n-stellig

$f(x_1, \dots, x_n) := f_1(x_1, \dots, x_n) \text{ op } f_2(x_1, \dots, x_n)$

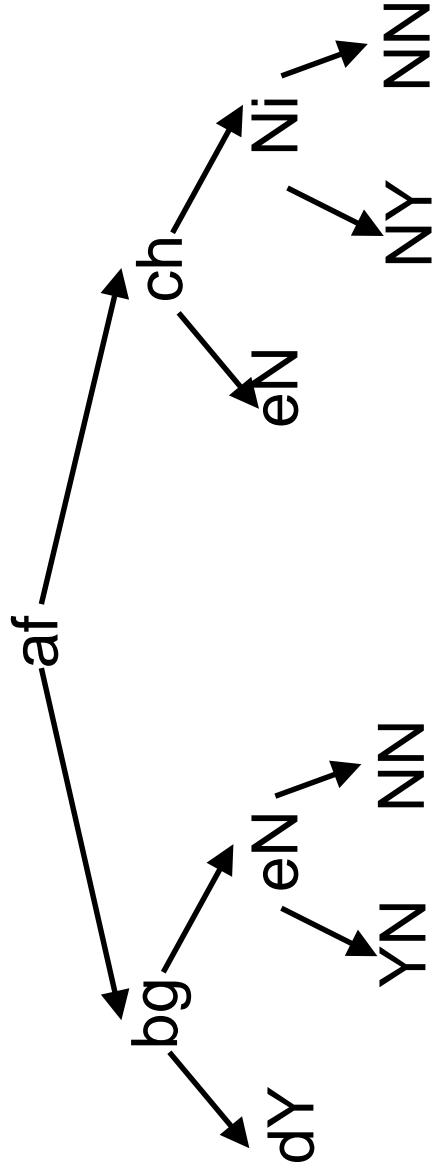
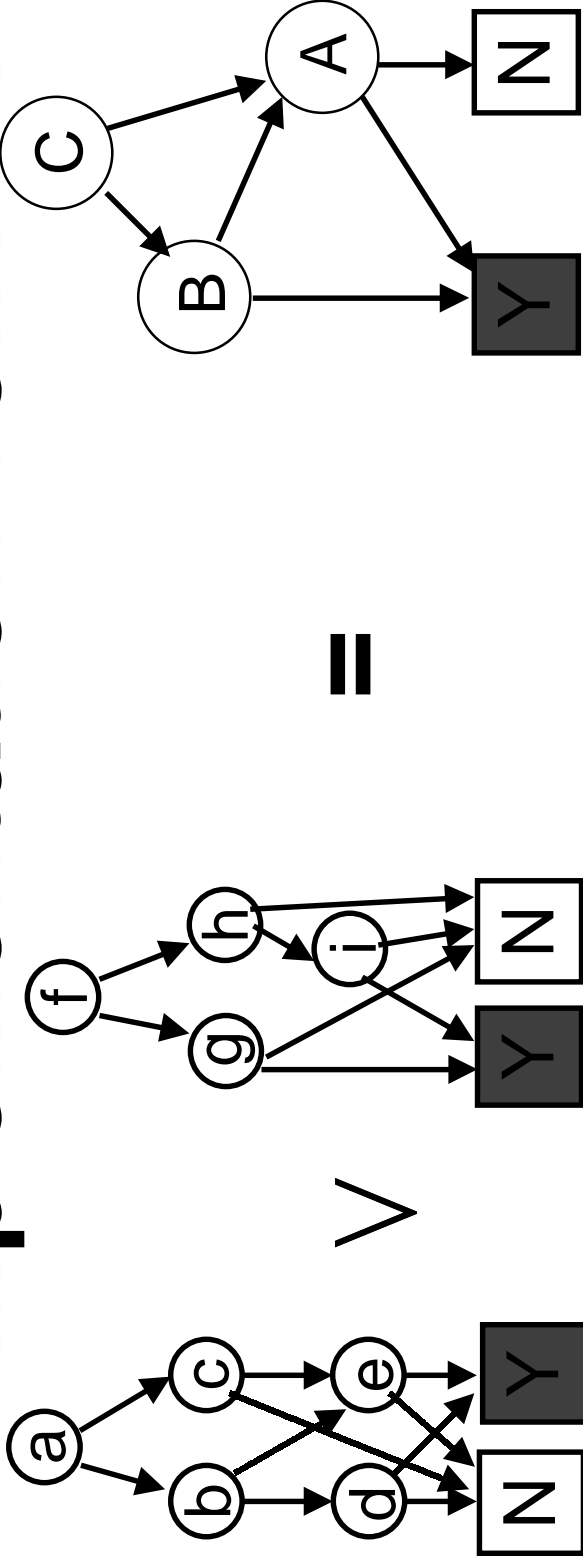
“Shannon-Expansion”

$$f(x_1, \dots, x_n) \Leftrightarrow (x_1 \wedge f(1, x_2, \dots, x_n)) \vee (\neg x_1 \wedge f(0, x_2, \dots, x_n))$$

$$x_1 \wedge (f_1(1, x_2, \dots, x_n) \text{ op } f_2(1, x_2, \dots, x_n)) \vee$$

$$\neg x_1 \wedge (f_1(0, x_2, \dots, x_n) \text{ op } f_2(0, x_2, \dots, x_n))$$

Implementation von APPLY



Y N A
 Y A B
 B A C

dY Y N A B Y A A C
 YN NN eN bg NY Ni ch af

$O(|BDD1| |BDD2|)$

RESTRICT

geg: f n -stellig b in $\{0,1\}$ i in $\{1, \dots, n\}$

$$f'(x_1, \dots, x_n) := f(x_1, \dots, x_{i-1}, b, x_{i+1}, \dots, x_n)$$

Implementation ähnlich zu APPLY

$$O(|BDD|)$$

Abgeleitete Operationen

$$\exists x_i . f(x_1, \dots, x_i, \dots, x_n) \quad f(x_1, \dots, 0, \dots, x_n) \vee f(x_1, \dots, 1, \dots, x_n)$$

(= 2 x RESTRICT, 1 x APPLY)

$$\begin{aligned} \text{Substitution: } h(x_1, \dots, x_n) &:= f(x_1, \dots, g(x_1, \dots, x_n), \dots, x_n) \\ &= g \wedge f(x_1, \dots, 1, \dots, x_n) \vee \neg g \wedge f(x_1, \dots, 0, \dots, x_n) \\ & \quad (= 2 \text{ x RESTRICT, } 3 \text{ x APPLY}) \end{aligned}$$

Mengenoperationen:

$U = \text{APPLY}(\quad , \vee)$ Komplement = tausche Y und N

$\cap = \text{APPLY}(\quad , \wedge)$

Relation = Menge von Paaren = BDD der Länge $2n$

$$R1 \circ R2: \exists y_1 \dots \exists y_n \quad f_1(x_1, \dots, x_n, y_1, \dots, y_n) \wedge f_2(y_1, \dots, y_n, z_1, \dots, z_n)$$

4.1.3 CTL Model Checking

Arbeiten mit $\text{SAT}_\phi = \{s \mid s \models \phi\}$ - repräsentiert als BDD

und $T(s,s')$ – der Zustandsübergangsrelation, auch als BDD

$T(s,s') = 1$ gdw. (s,s') in E

$T(s,s')$ kann aus einer Systembeschreibung als nextstate-Funktion generiert werden

System erfüllt ϕ gdw. $I \subseteq \text{SAT}_\phi$

Beispiel:3-bit-Zähler

$$\begin{aligned} &x_0' \Leftrightarrow \neg x_0 \\ &\wedge \\ &x_1' \Leftrightarrow (\neg x_1 \Leftrightarrow x_0) \\ &\wedge \\ &x_2' \Leftrightarrow (\neg x_2 \Leftrightarrow (x_0 \wedge x_1)) \end{aligned}$$

CTL Model Checking – einfache Fälle

1. Elementare Aussagen – setzen BDD voraus, normalerweise kein Problem
2. $\phi \wedge \psi$ $\phi \vee \psi$ $\neg \phi$ -- einfache Anwendung von APPLY
3. $\text{SAT}_{\text{EX } \phi} \Leftrightarrow \exists y_1 \dots \exists y_n [\text{T}(x_1, \dots, x_n, y_1, \dots, y_n) \wedge \text{SAT}_{\phi}(y_1, \dots, y_n)]$
4. $\text{AX } \phi \Leftrightarrow \neg \text{EX } \neg \phi$

bleiben: EU und EG – Rest ist dann ableitbar per Tautologien

Model Checking EU

geg: SAT_ϕ , SAT_ψ ges: $SAT_{E(\phi \cup \psi)}$

wissen: $SAT_\psi \subseteq SAT_{E(\phi \cup \psi)}$

wissen auch: Wenn s' in $SAT_{E(\phi \cup \psi)}$ und s in SAT_ϕ und $T(s, s')$,
so ist auch s in $SAT_{E(\phi \cup \psi)}$

und schließlich: Weiter nix

$Z := SAT_\psi$
do
$Z := Z \vee (SAT_\phi \wedge SAT_{EX Z})$
until nothing changes
$SAT_{E(\phi \cup \psi)} := Z$

Berechnen kleinsten Fixpunkt eines monoton wachsenden
Mengenoperators

Model Checking EG

geg: SAT_ϕ

ges: $SAT_{EG\phi}$

wissen: $SAT_{EG\phi} \subseteq SAT_\phi$

wissen auch: Wenn s' in $SAT_{EG\phi}$ und s in SAT_ϕ und $T(s,s')$,
so ist auch s in $SAT_{EG\phi}$

und schließlich: Weiter nix

$Z := SAT_\phi$
do
$Z := Z \wedge SAT_{EXZ}$
until nothing changes
$SAT_{EG\phi} := Z$

Berechnen größten Fixpunkt eines monoton fallenden
Mengenoperators

By the way....

- μ-Kalkül ≈ einfache Aussagen (Mengen)
- + boolesche Operationen (Mengenoperationen)
- + “kleinster Fixpunkt von ...“
- + “größter Fixpunkt von ...”

wird auch als Spezifikationsprache verwendet

Bottlenecks

1. Größe des BDD für Z in den Fixpunktberechnungen

Lösung: Gelegentliches Umordnen der Variablen,
z.B. Sifting = tausche 2 benachbarte Variablen und
schaue, ob BDD dadurch kleiner wird

2. Das BDD für die Übergangsrelation T

Lösung: Partitionierung von T

Partitionierung der Übergangsrelation

Idee: T ist meistens Konjunktion Teilformeln

Beispiel: $T1: x0' \Leftrightarrow \neg x0$

$T2: x1' \Leftrightarrow (\neg x1 \Leftrightarrow x0)$

$T3: x2' \Leftrightarrow (\neg x2 \Leftrightarrow (x0 \wedge x1))$

$T \Leftrightarrow T1 \wedge T2 \wedge T3$

Partitionen kleiner als T , günstigenfalls auch in der Summe

mindestens: Eine Partition hängt normalerweise nicht von allen Variablen ab, ist also auf jeden Fall flacher als das BDD von T

Nutzt das?

Frühe Quantifizierung

T wird verwendet für EX:

$$\exists Y [T(X, Y) \wedge SAT_{\phi}(Y)]$$

partitioniert: $\exists Y [T_1(X, Y) \wedge \dots \wedge T_n(X, Y) \wedge SAT_{\phi}(Y)]$

Normalerweise sind \exists und \wedge nicht vertauschbar, es sei denn, in einer Seite von \wedge kommt die quantifizierte Variable nicht vor

$$Y_i := \{ y \text{ in } Y \mid y \text{ kommt nicht in } T_1 \dots T_{i-1} \text{ vor} \}$$

→

$$\exists Y_1(T_1(X, Y) \wedge \exists Y_2 (T_2(X, Y) \wedge (\dots \wedge \exists Y_n (T_n(X, Y) \wedge SAT_{\phi}(Y)) \dots)))$$

Effekt: BDD von $\exists Y_i T_i(X, Y)$ hängt nicht von Variablen in Y_i ab, ist also flacher als das von BDD von $T_i(X, Y)$
→ kleinere BDDs in Zwischenschritten!!