Gültigkeit und Erfüllbarkeit

F gilt in $\mathcal{A}$ ($\mathcal{A}$ ist Modell von F):

$$\mathcal{A} \models F \;:\Leftrightarrow\; \mathcal{A}(F) = 1$$

F ist (allgemein-) gültig (oder eine Tautologie):

$$\models F \;:\Leftrightarrow\; \mathcal{A} \models F, \text{ für alle } A : \Pi \to \{0,1\}$$

F heißt erfüllbar gdw. es $\mathcal{A} : \Pi \to \{0,1\}$ gibt, so daß $\mathcal{A} \models F$.

Sonst heißt F unerfüllbar (oder eine Kontradiktion).

Folgerung und Äquivalenz

F impliziert G (oder G folgt aus F), i.Z. $F \models G$

$$:\Leftrightarrow \quad \text{für alle } \mathcal{A} : \Pi \to \{0,1\} \text{ gilt: } \mathcal{A} \models F \Longrightarrow \mathcal{A} \models G.$$

F und G sind äquivalent

$$:\Leftrightarrow \quad \text{für alle } \mathcal{A} : \Pi \to \{0,1\} \text{ gilt: } \mathcal{A} \models F \text{ gdw. } A \models G.$$

Erweiterung auf Formelmengen N in natürlicher Weise, z.B.:

$N \models G \quad :\Leftrightarrow \quad$ für alle $\mathcal{A} : \Pi \to \{0,1\}$ gilt:

falls $\mathcal{A} \models F$, für alle $F \in N$,

so $\quad \mathcal{A} \models G$.

Unerfüllbarkeitstest

Jede Formel F enthält endlich viele Aussagenvariablen.
$\mathcal{A}(F)$ ist nur von der Werten dieser Aussagenvariablen abhängig.

F enthält n Aussagenvariablen:

$\Rightarrow$ 2^n Wertbelegungen notwendig um zu überprüfen, ob F erfüllbar ist oder nicht.

$\Rightarrow$ Wahrheitstafel

$\Rightarrow$ Das Erfüllbarkeitsproblem ist entscheidbar
(Cook's Theorem: NP-vollständig)

Es existieren viel bessere Methoden als Wahrheitstafeltests um die Erfüllbarkeit einer Formel zu überprüfen.

Strukturelle Induktion

Induktive Definition von Formeln:

- es werden zunächst die einfachsten Formeln definiert;
- dann wird erklärt, wie aus einfachen Formeln kompliziertere Formeln aufgebaut werden können.

Beweise "$\mathcal{B}(F)$ gilt für alle Formeln F".

Strukturelle Induktion (Induktion über den Formelaufbau)

Induktionsbasis: Man zeige, es gilt $\mathcal{B}(P)$ für alle $P \in \Pi$

Induktionsschritt: (Induktions-)Annahme: $\mathcal{B}(F_1), \mathcal{B}(F_2)$ gelten
Man zeige, unter dieser Annahme, dass
$\mathcal{B}(\neg F_1)$, $\mathcal{B}(F_1 \wedge F_2)$, $\mathcal{B}(F_1 \vee F_2)$ gelten.

Substitutionslemma

Satz 1.5 *Seien F und G äquivalente Formeln. Sei H eine Formel mit (mindestens) einem Vorkommen der Teilformel F.*

Dann ist H äquivalent zu H', wobei H' aus H hervorgeht, indem (irgend) ein Vorkommen von F in H durch G ersetzt wird.

Beweis: Strukturelle Induktion.

Wichtige Äquivalenzen

Die folgenden Äquivalenzen sind für alle Formeln F, G, H gültig:

$$
\begin{aligned}
(F \wedge F) &\equiv F \\
(F \vee F) &\equiv F && \text{(Idempotenz)} \\[1ex]
(F \wedge G) &\equiv (G \wedge F) \\
(F \vee G) &\equiv (G \vee F) && \text{(Kommutativität)} \\[1ex]
(F \wedge (G \wedge H)) &\equiv ((F \wedge G) \wedge H) \\
(F \vee (G \vee H)) &\equiv ((F \vee G) \vee H) && \text{(Assoziativität)} \\[1ex]
(F \wedge (F \vee G)) &\equiv F \\
(F \vee (F \wedge G)) &\equiv F && \text{(Absorption)} \\[1ex]
(F \wedge (G \vee H)) &\equiv ((F \wedge G) \vee (F \wedge H)) \\
(F \vee (G \wedge H)) &\equiv ((F \vee G) \wedge (F \vee H)) && \text{(Distributivität)}
\end{aligned}
$$

Wichtige Äquivalenzen

Die folgenden Äquivalenzen sind für alle Formeln F, G, H gültig:

$$(\neg\neg F) \equiv F \qquad \text{(Doppelte Negation)}$$

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

$$\neg(F \vee G) \equiv (\neg F \wedge \neg G) \qquad \text{(De Morgan's Regeln)}$$

$$(F \wedge G) \equiv F, \text{ falls } G \text{ Tautologie}$$

$$(F \vee G) \equiv \top, \text{ falls } G \text{ Tautologie} \qquad \text{(Tautologieregeln)}$$

$$(F \wedge G) \equiv \bot, \text{ falls } G \text{ unerfüllbar}$$

$$(F \vee G) \equiv F, \text{ falls } G \text{ unerfüllbar} \qquad \text{(Tautologieregeln)}$$

1.4 Binäre Entscheidungsdiagramme

- Darstellung für Boole'sche Funktionen

- Effizient, z.B. für Gleichheitstests

1.4 Binäre Entscheidungsdiagramme

- Darstellung für Boole'sche Funktionen

- Effizient, z.B. für Gleichheitstests

Boole'sche Funktionen

$f : \{0,1\}^n \to \{0,1\}$ Boole'sche Funktion.

Anwendungen Hardware von Computern:

Schaltkreise, die Boole'sche Funktionen berechnen

Boole'sche Funktionen

$f : \{0,1\}^n \to \{0,1\}$ Boole'sche Funktion.

Beschreibung

- Wahrheitstabellen

  Speicher-ineffizient (n Variablen $\mapsto 2^n$ Zeilen)
  Erfüllbarkeitstest: gibt es einen 1 in der letzten Spalte?
  Gleichheit von Funktionen: letzte Spalte identisch?

- Formeln

  Kompakte und effiziente Repräsentation
  Erfüllbarkeitstest, Gleichheitstest: NP-vollständig

- Spezielle Klassen von Formeln (DNF, KNF)

Boole'sche Funktionen

Beschreibung	kompakt?	Test		Operationen		
		Erfüll.	Gültig	$\wedge$	$\vee$	$\neg$
Formeln	oft	schwer	schwer	leicht	leicht	leicht
DNF	manchmal	leicht	schwer	schwer	leicht	schwer
KNF	manchmal	schwer	leicht	leicht	schwer	schwer
Wh.tabelle	nicht	schwer	schwer	schwer	schwer	schwer

Boole'sche Funktionen

Beschreibung von Boole'sche Funktionen durch Formeln

- $F(P_1, \dots, P_n)$ Formel $\quad \mapsto \quad f_F : \{0,1\}^n \to \{0,1\}$

- $f : \{0,1\}^n \to \{0,1\} \quad \mapsto \quad F$ Formel mit $f_F = f$.

Formeln und Boole'sche Funktionen

$\{0,1\}^\Pi := \{\mathcal{A} \mid \mathcal{A} : \Pi \to \{0,1\}\}$ Menge aller Valuationen.

$$F \in F_\Pi \qquad \mapsto \qquad f_F : \{0,1\}^\Pi \to \{0,1\}$$
$$f_F(\mathcal{A}) := \mathcal{A}(F).$$

$F \in F_\Pi$ enthält endlich viele Aussagenvariablen $\{P_1, \ldots, P_n\}$

$\mapsto \qquad f_F : \{0,1\}^n \to \{0,1\}$

$f_F(a_1, \ldots, a_n) := \mathcal{A}(F)$, wo $\mathcal{A}(P_i) = a_i$ for $i = 1, \ldots, n$

Boole'sche Funktion assoziiert mit F

Formeln und Boole'sche Funktionen

Satz 1.6 *Seien $F(P_1, \ldots, P_n)$ und $G(P_1, \ldots, P_n)$ äquivalent. Dann sind die assoziierten Boole'sche Funktionen gleich.*

Beispiel: $F(P, Q) = P \wedge (Q \vee \neg P)$ und $G(P, Q) = P \wedge Q$ haben die gleiche assoziierte Funktion.

Bemerkung: Die Ordnung der Variablen ist wichtig.

$F(P, Q) = P \wedge (Q \vee P)$ und $G(Q, P) = P \wedge (Q \vee P)$

haben verschiedenen assoziierten Funktionen.

Boole'sche Funktionen und Formeln

$f : \{0,1\}^n \to \{0,1\}, \quad \{P_1, \ldots, P_n\} \subseteq \Pi$

$$F_f := \bigvee_{(a_1,\ldots,a_n)\in\{0,1\}^n} (f(a_1, \ldots, a_n) \wedge P_1^{a_1} \wedge \cdots \wedge P_n^{a_n})$$

wo $P^1 := P$ und $P^0 := \neg P$

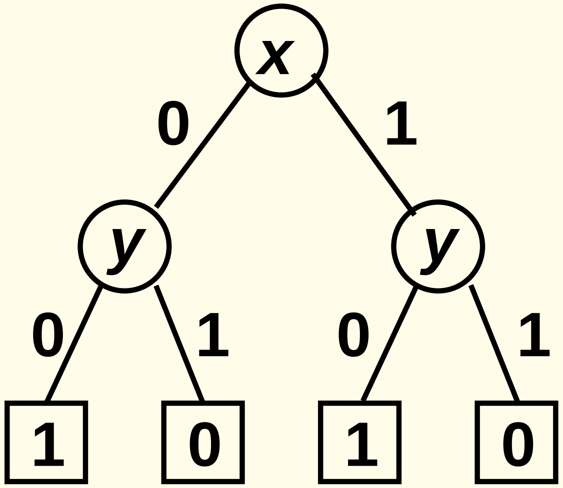
Satz 1.7 *Die Boole'sche Funktionen f_{F_f} und f sind gleich, i.e.*

für alle $a_1, \ldots, a_n \in \{0,1\}$

$$f_{F_f}(a_1, \ldots, a_n) := f(a_1, \ldots, a_n)$$

Binäre Entscheidungsbäume

Geordnete Binärbäume, deren Blätter mit 0 oder 1
und deren andere Knoten mit Variablen markiert sind.

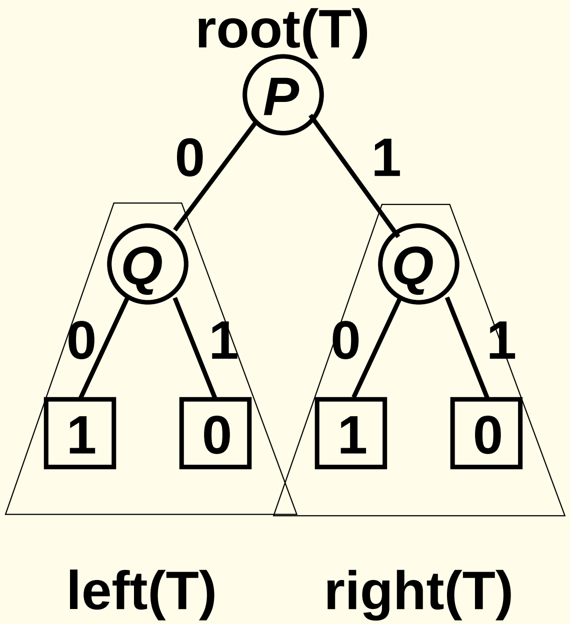


Binäre Entscheidungsbäume

Definition Sei T endlicher geordneter binärer Entscheidungsbaum mit Variablen $\{P_1, \ldots, P_n\}$.
Zu T entspricht eine (eindeutig bestimmte) Boole'sche Funktion $f_T : \{0, 1\}^n \to \{0, 1\}$:

Binäre Entscheidungsbäume

Definition Sei T endlicher geordneter binärer Entscheidungsbaum mit Variablen $\{P_1, \ldots, P_n\}$.
Zu T entspricht eine (eindeutig bestimmte) Boole'sche Funktion $f_T : \{0,1\}^n \to \{0,1\}$:



Sei $\mathcal{A} : \{P_1, \ldots, P_n\} \to \{0,1\}$, mit $\mathcal{A}(P_i) = a_i$

P markiert die Wurzel von T:

falls $\mathcal{A}(P) = 0$: $\quad f_T(\overline{a}) := f_{\mathrm{left}(T)}(\overline{a})$

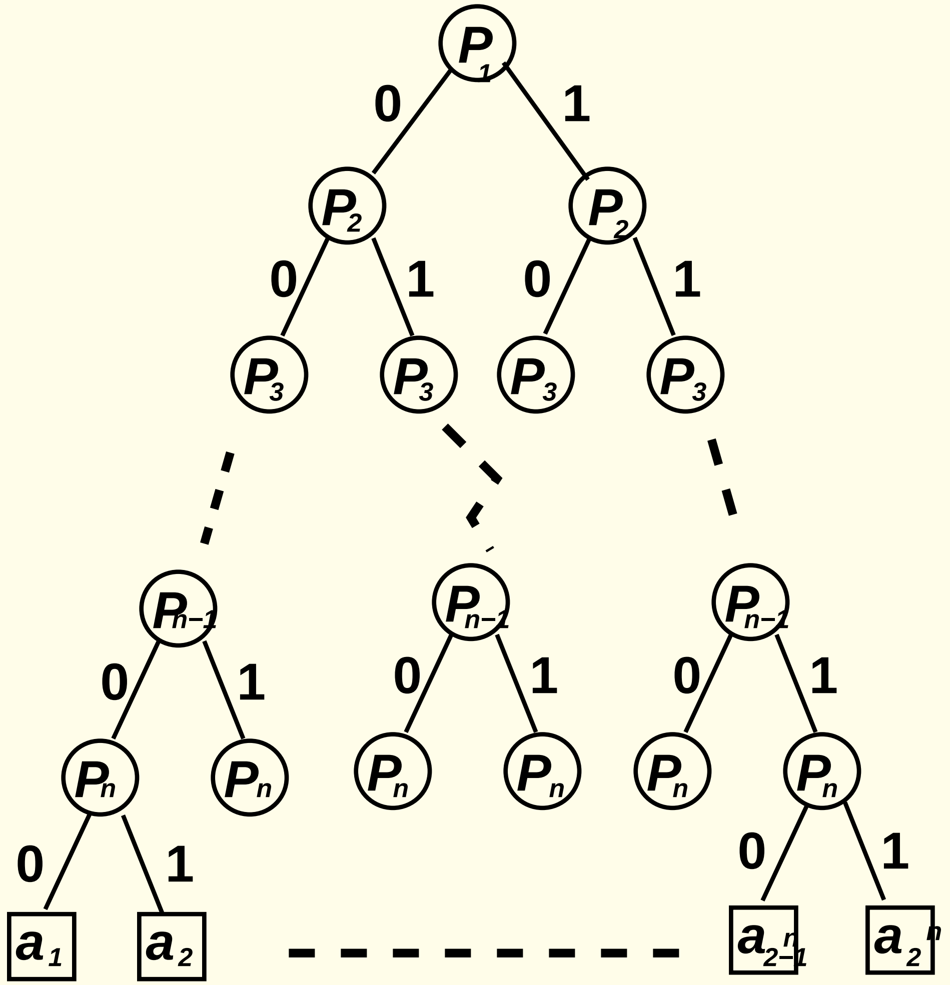
falls $\mathcal{A}(P) = 1$: $\quad f_T(\overline{a}) := f_{\mathrm{right}(T)}(\overline{a})$

0 markiert die Wurzel von T: $\quad f_T(\overline{a}) := 0$

1 markiert die Wurzel von T: $\quad f_T(\overline{a}) := 1$

Binäre Entscheidungsbäume

$f : \{0,1\}^n \to \{0,1\} \quad \mapsto$



$f(0\ldots0)$ $f(0\ldots1)$ $\ldots$ $f(v_1\ldots v_n)$ $\ldots$ $f(1\ldots0)$ $f(1\ldots1)$

Entscheidungsbäume und Wahrheitstabellen

$f : \{0,1\}^n \to \{0,1\} \quad \mapsto \quad$ Entscheidungsbaum mit mindestens $2^{n+1} - 1$ Knoten.

- genau so ineffizient wie Wahrheitstabellen
- redundante Information $\mapsto$ Optimierung möglich

z.B. 1. Entferne duplizierte Blätter

2. Entferne überflüssige Tests

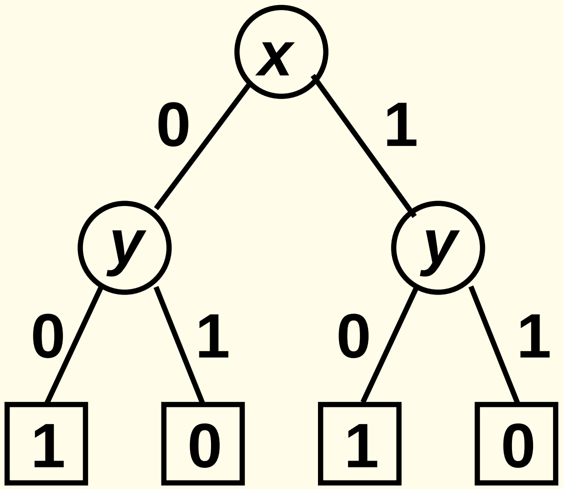
3. Entferne duplizierte Knoten

$\mapsto$ Binäre Entscheidungsdiagramme (BDD)

Minimale Graphdarstellung

1. Entferne duplizierte Blätter

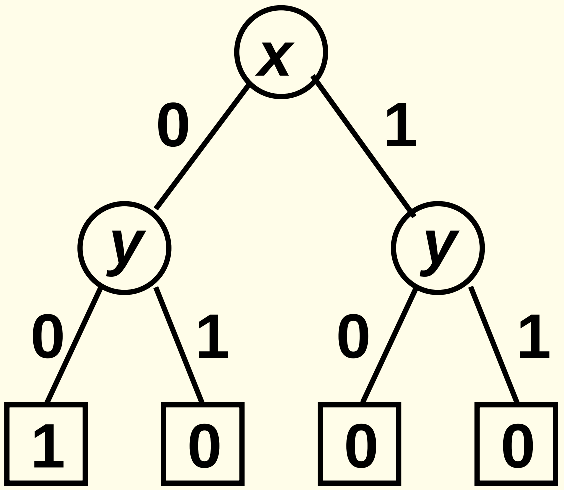
   - nur eine Kopie von 0, 1 notwendig

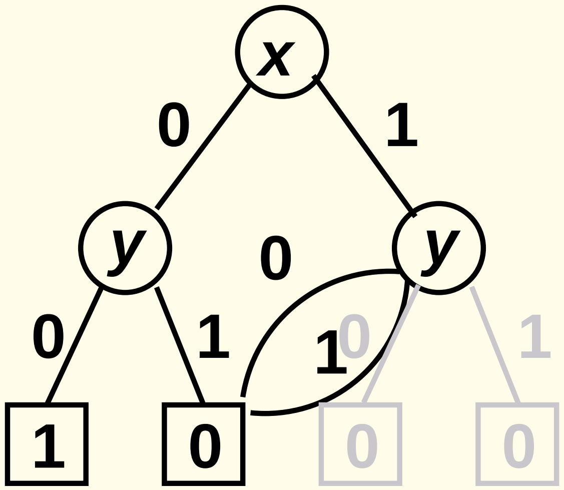


Minimale Graphdarstellung

1. Entferne duplizierte Blätter

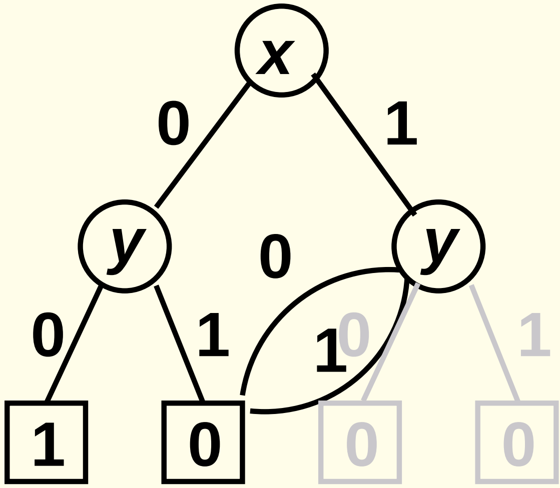
   - nur eine Kopie von 0, 1 notwendig





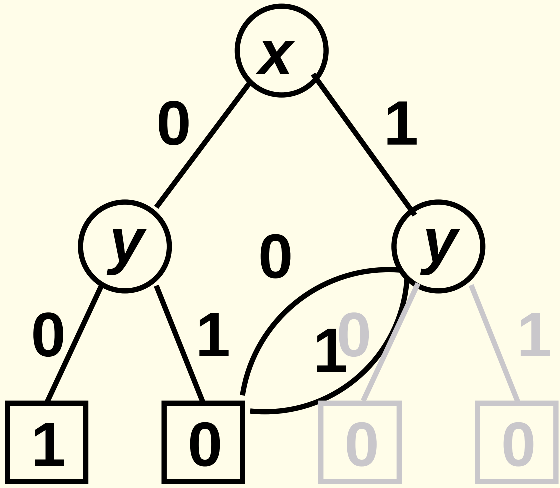
Minimale Graphdarstellung

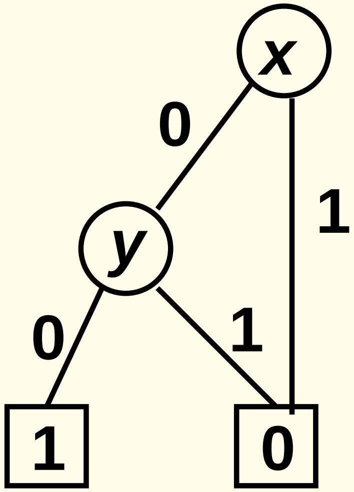
2. Entferne überflüssige Tests



Minimale Graphdarstellung

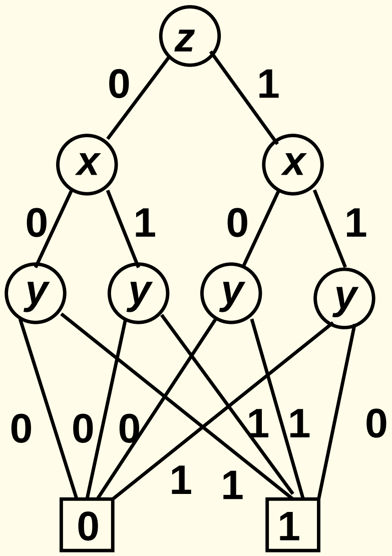
2. Entferne überflüssige Tests





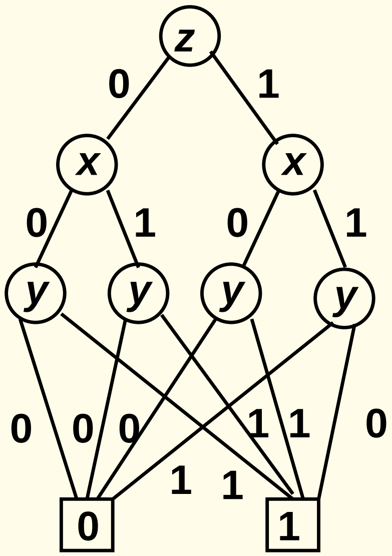
Minimale Graphdarstellung

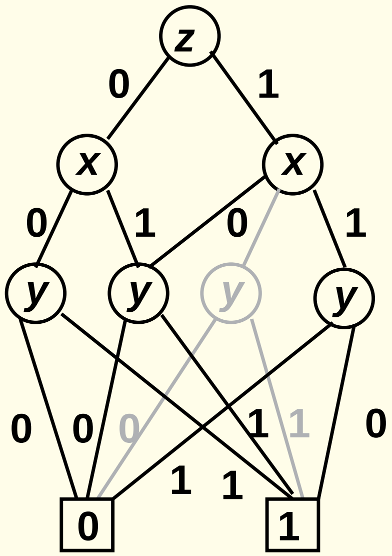
3. Entferne duplizierte nichtterminale Knoten



Minimale Graphdarstellung

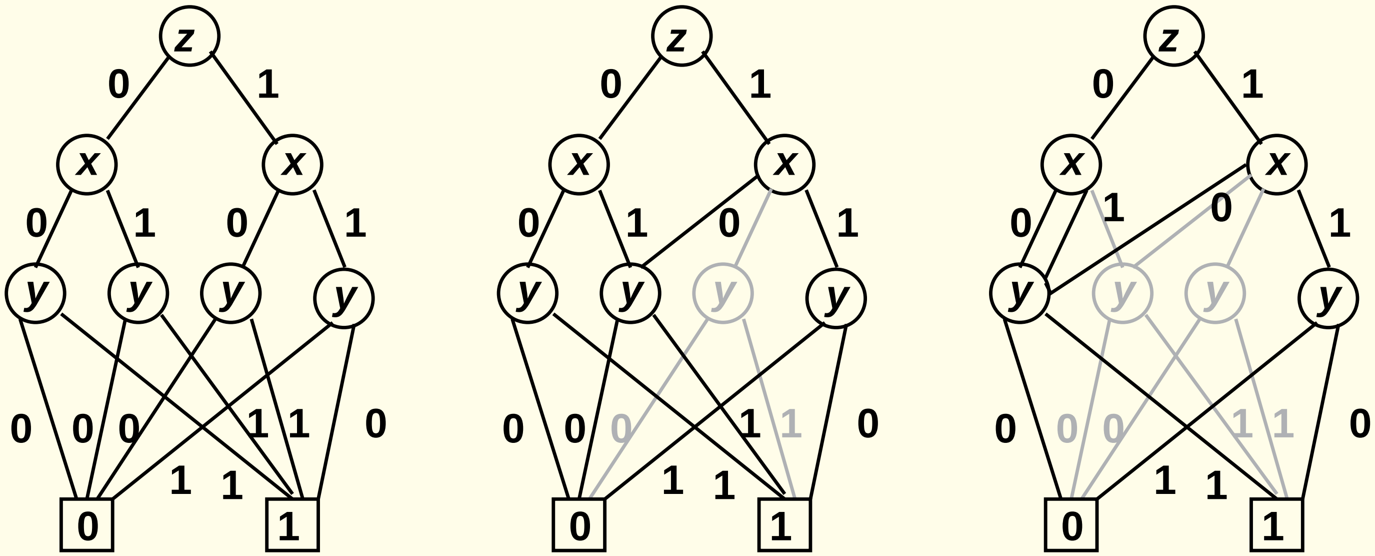
3. Entferne duplizierte nichtterminale Knoten





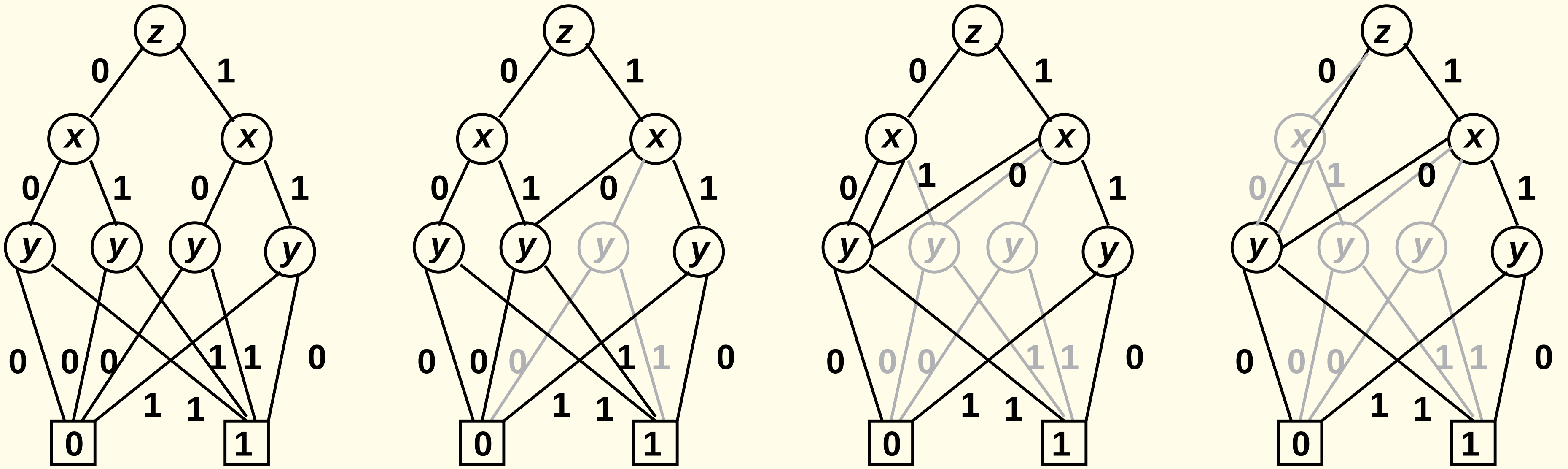
Minimale Graphdarstellung

3. Entferne duplizierte nichtterminale Knoten



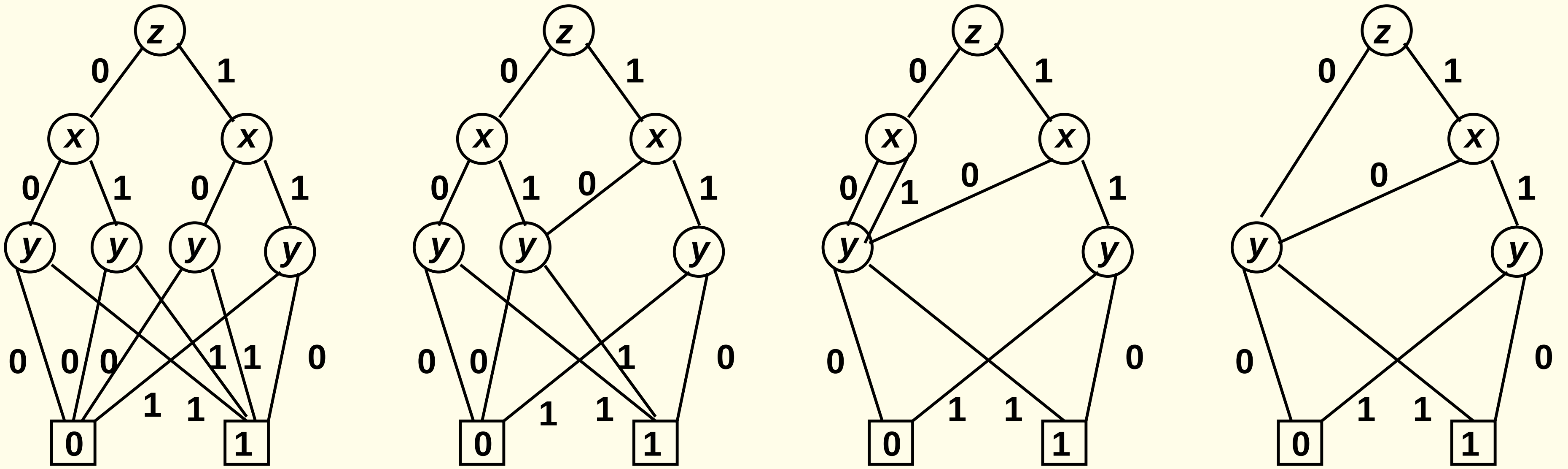
Minimale Graphdarstellung

3. Entferne duplizierte nichtterminale Knoten



Minimale Graphdarstellung

3. Entferne duplizierte nichtterminale Knoten



Binäre Entscheidungsdiagramme (BDDs)

gerichteter Graph: $(G, \rightarrow), \quad \rightarrow \subseteq G \times G$

Zyklus in gerichtetem Graph: Pfad $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$

gerichteter azyklischer Graph (DAG): gerichteter Graph ohne Zyklen

Binäres Entscheidungsdiagramm (BDD) endlicher DAG:

- einen einzigen initiellen Knoten
- terminale Knoten mit 0 oder 1 markiert
- nichtterminalen Knoten mit einer Aussagenvariable markiert
- jeder nichtterminaler Knoten: genau 2 Kanten (markiert 0/1)

Reduziertes binäres Entscheidungsdiagramm

Optimierungen 1–3 können nicht angewandt werden.

Binäre Entscheidungsdiagramme (BDDs)

Problem Variablen können mehrmals auf einem Pfad erscheinen.

(Operationen mit BDDs)

Binäre Entscheidungsdiagramme (BDDs)

Problem Variablen können mehrmals auf einem Pfad erscheinen.

(Operationen mit BDDs)

Lösung Geordnete BDDs

Geordnete Entscheidungsdiagramme (OBDDs)

$[P_1, \dots, P_n]$ geordnete Liste von Variablen (ohne Wiederholungen)

Sei B ein BDD in Variablen $\{P_1, \dots, P_n\}$

B hat die Ordnung $[P_1, \dots, P_n]$
wenn für jedes Pfad $v_1 \to v_2 \to \cdots \to v_m$ in B,

falls - $i < j$,

- die Markierung von v_i ist P_{k_i}

- die Markierung von v_j ist P_{k_j}

so $k_i < k_j$.

Ein geordneter BDD (i.Z. OBDD) ist ein BDD, der eine Ordnung hat, für eine bestimmte geordnete Liste von Variablen.