Mathematik IT-1

Diskrete Mathematik für Informatik-Studiengänge

10. Januar 2022

Prof. G. Averkov

Institut für Mathematik, Fakultät 1

Fachgebiet Algorithmische Mathematik

Brandenburgische Technische Universität Cottbus-Senftenberg

Inhaltsverzeichnis

Kapitel V

Boolesche Algebra

1 Boolesche Funktionen und Formeln

1.1 Def. Eine **boolesche Variable** ist eine Variable mit den Werten aus

$$\{\textsc{Falsch}, \textsc{Wahr}\}.$$

1.2. Wir benutzen die Standard-Identifizierung $\{\textsc{Falsch}, \textsc{Wahr}\} \to \{0, 1\}$ mit

$$\textsc{Falsch} \mapsto 0 \qquad\qquad \textsc{Wahr} \mapsto 1,$$

welche uns ermöglicht, die kürzere Schreibweise mit 0 und 1 zu nutzen.

1.3 Def. Für $n \in \mathbb{N}$ nennen wir $f : \{0,1\}^n \to \{0,1\}$ **boolesche Funktion** in n Variablen.

1.4. Es gibt genau vier boolesche Funktionen in einer Variablen:

x	0	x	$\overline{x}$	1
0	0	0	1	1
1	0	1	0	1

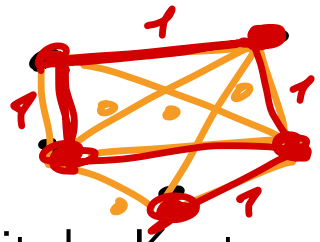
1.5 Bsp.

(a) Die **Mehrheitsfunktion**:

$$f(x_1, \ldots, x_n) := \begin{cases} 1 & \text{wenn mindestens } n/2 \text{ der Variablen} \\ & x_1, \ldots, x_n \text{ gleich } 1 \text{ gesetzt sind,} \\ 0 & \text{sonst.} \end{cases}$$

(b) Die **Parity-Funktion**:

$$f(x_1, \ldots, x_n) := \begin{cases} 1 & \text{wenn ungerade viele Variablen} \\ & x_1, \ldots, x_n \text{ gleich } 1 \text{ gesetzt sind,} \\ 0 & \text{sonst.} \end{cases}$$



(c) Sei V endliche Menge. Dann lässt sich jeder Graph $G = (V, E)$ mit der Knotenmenge V als die characteristische Funktion $1_E \in \{0,1\}^{\binom{V}{2}}$ der Menge der Kanten kodieren. Verscheidene Eigenschaften von Graphen mit der Knotenmenge V können dann als boolesche Funktionen auf $\{0,1\}^{\binom{V}{2}}$ beschrieben weren. Etwa

$$f(x) := \begin{cases} 1 & \text{der Graph } (V,E) \text{ mit } 1_E = x \\ & \text{ist zusamenhängend,} \\ 0 & \text{sonst.} \end{cases}$$

(d) Der Operator

$$x \,?\, y \,:\, z = \begin{cases} y, & \text{für } x = 1, \\ z, & \text{für } x = 0, \end{cases}$$

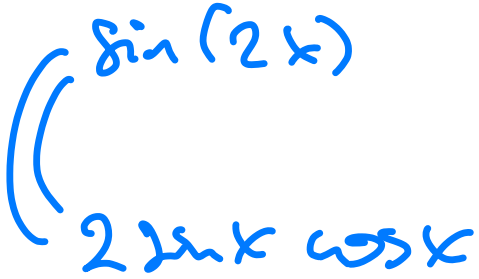
der in C/C++ und Java verfügbar ist, ist im Fall $y, z \in \{0,1\}$ eine boolesche Funktion

in drei Variablen.

1.6 Def. Eine **boolesche Formel** ist ein Ausddruck der aus endlich vielen booleschen Variablen, den Verknüpfungen $\neg$, $\vee$, $\wedge$ und Konstanen $0, 1$ konstruiert ist. Eine boolesche Formel, die auf den Variablen $x_1, \ldots, x_n$ $(n \in \mathbb{N})$ definiert ist, ergibt durch das Auswerten die boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ zu dieser Formel.

Zwei Formeln nennen wir **äquivalent**, wenn sie dieselbe boolesche Funktion definieren.

In den booleschen Formel schreiben wir oft auch $a \cdot b$ oder ab an der Stelle von $a \wedge b$ und setzten dabei die Priorität von $\cdot$ höher als die Priorität von $\vee$.



1.7 Def. Die selber boolesche Funktion kann durch verschiedene Formeln gegeben werde. Etwa,

$$x\,\overline{y} \vee y = x \vee y.$$

1.8. Wenn zwei Formeln F und G äquivalent sind, so bezeichnet man das in manchen Quellen als $F \equiv G$, wir nutzen aber hier (wie in [Lov20]) die Bezeichnung $F = G$ (die wir ja auch sonst, etwa für die Gleichheit von Funktionen benutzen).

1.9 Def. Wir nennen eine boolesche Formel in n Variablen **erfüllbar**, wenn sie eine eine boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ definiert, die nicht identisch gleich 0 ist. Das bedeutet, dass die Formel bei einer Belegung von Variablen zu 1 ausgewertet wird.

1.10 Def. Wir nennen eine boolesche Formel in n Variablen eine **Tautologie**, wenn sie eine boolsche Funktion $f : \{0,1\}^n \to \{0,1\}$ definiert, die identisch gleich 1 ist. Das bedeutet, dass die Formel bei jeder Belegung von Variablen zu 1 ausgewertet wird.

1.11 Prop. Für alle $a, b, c \in \{0, 1\}$ gelten die folgenden Gleichungen:

$a \wedge b = b \wedge a$	$a \vee b = b \vee a$	Kommutativgesetze
$(a \wedge b) \wedge c = a \wedge (b \wedge c)$	$(a \vee b) \vee c = a \vee (b \vee c)$	Distributivgesetze
$a \wedge a = a$	$a \vee a = a$	Idempotenzgesetze
$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	Distributivgesetze
$a \wedge 1 = a$	$a \vee 0 = a$	Neutralitätsgesetze
$a \wedge 0 = 0$	$a \vee 1 = 1$	Extremalgesetze
$\overline{\overline{a}} = a$		Doppelnegationsgesetz
$\overline{a \wedge b} = \overline{a} \vee \overline{b}$	$\overline{a \vee b} = \overline{a} \wedge \overline{b}$	De Morgansche Gesetze
$a \wedge \overline{a} = 0$	$a \vee \overline{a} = 1$	Komplementärgesetze
$\overline{0} = 1$	$\overline{1} = 0$	Dualitätsgesetze
$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$	Absorptionsgesetze

2 Darstellung durch DNF

2.1 Def. Ist x eine boolesche Variable, so nennen wir die Formeln x und $\overline{x}$ **Literale**.

Wir nennen Disjunktion endlich vieler Literale eine **Elementardisjunktion** (kurz **ED**) oder eine **Klausel** und Konjunktion endlich vieler Literale eine **Elementarkonjunktion** (kurz **EK**). Die 0 ist dabei die Disjunktion von 0 Literalen und die 1 die Konjunktion von 0 Literalen.

Des Weiteren nennen wir Konjunktion endlich vieler EDs eine **konjunktive Normalform** (kurz **KNF**) und Disjunktion endlich vieler EDs eine **disjunktive Normalform** (kurz **DNF**).

$\overline{x} \vee y \vee \overline{z}$

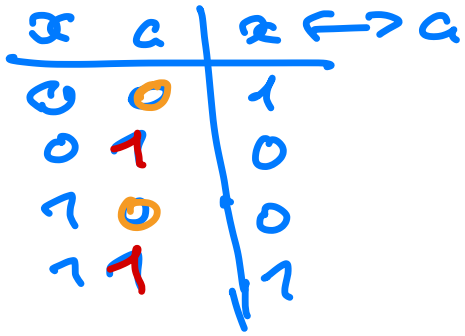
2.2 Lem. Für alle $x, a \in \{0,1\}$ gilt

$$x \leftrightarrow a = \begin{cases} x & \text{für } a = 1, \\ \overline{x} & \text{für } a = 0 \end{cases} \tag{V.1}$$

und

$$\overline{x \leftrightarrow a} = x \leftrightarrow \overline{a}. \tag{V.2}$$

Beweis. Man kann alle vier Möglichkeiten für $(x, a) \in \{0,1\}^2$ durchprobieren, um die Gleichheit zu verifizieren. □



2.3 Thm. *Jede boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ kann durch eine DNF beschrieben werden.*

Beweis. Wir betrachten die Belegungen von $(x_1, \ldots, x_n)$, auf denen f zu 1 ausgewertet wird: sei

$$W := \{(a_1, \ldots, a_n) \in \{0,1\}^n \,:\, f(a_1, \ldots, a_n) = 1\}\,.$$

Es gilt:

$$f(x_1, \ldots, x_n) = \bigvee_{(a_1,\ldots,a_n)\in W} \bigwedge_{i=1}^{n} (x_i \leftrightarrow a_i). \tag{V.3}$$

Um das zu sehen argumentieren wir wie folgt. Für alle $x_1, \ldots, x_n \in \{0, 1\}$ gilt:

$x_i = a_i$ heißt $(x_i \leftrightarrow a_i) = 1$

$$
\begin{aligned}
& f(x_1, \ldots, x_n) \text{ ist wahr} \\
\Leftrightarrow \quad & (x_1, \ldots, x_n) \in W \\
\Leftrightarrow \quad & \text{es gibt ein } (a_1, \ldots, a_n) \in W \text{ mit } x_i = a_i \text{ für alle } i = 1, \ldots, n \\
\Leftrightarrow \quad & \text{es gibt ein } (a_1, \ldots, a_n) \in W, \text{ für welches } \bigwedge_{i=1}^{n} (x_i \leftrightarrow a_i) \text{ wahr ist} \\
\Leftrightarrow \quad & \bigvee_{(a_1,\ldots,a_n) \in W} \bigwedge_{i=1}^{n} (x_i \leftrightarrow a_i) \text{ ist wahr.}
\end{aligned}
$$

Wegen (V.1) ist der Ausdruck $x_i \leftrightarrow a_i$ in (V.3) je nach dem Wert von a_i entweder x_i oder $\overline{x_i}$. Mit dieser Interpretation ist die rechte Seite von (V.3) eine DNF. □

2.4. (V.3) ist ein Rezept der Konstruktion einer Darstellung als DNF für ein gegebenes f. Für jede Belegung $(a_1, \ldots, a_n)$, bei der f wahr wird, füge die Elementarkonjunktion hinzu deren Literale für jedes $i = 1, \ldots, n$ folgendermaßen gewählt werden: x_i bei $a_i = 1$ und $\overline{x_i}$ bei $a_i = 0$.

Bsp.

Man finde eine Darstellung von f als eine DNF.

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$$f = \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3}$$

Ist $f = x_1 \dot{\vee} x_3$?

$f(1,1,0) = 0$
$1 \dot{\vee} 0 = 1$

Theorem Felix $\Longrightarrow$

$f \neq x_1 \dot{\vee} x_3$

Bsp.

x	y	$x \dot{\vee} y$	$x \leftrightarrow y$	$x \rightarrow y$
0	0	0	1	1
0	1	1	0	1
1	0	1	0	0
1	1	0	1	1

$$x \dot{\vee} y = \bar{x} \cdot y \vee x \cdot \bar{y}$$

$$x \leftrightarrow y = \bar{x} \cdot \bar{y} \vee x \cdot y$$

$$x \rightarrow y = \bar{x} \cdot \bar{y} \vee \bar{x} \cdot y \vee x \cdot y$$

kann man eine einfachere DNF aus Darstellung von $x \rightarrow y$ finden?

$\bar{x} \cdot y \vee x \cdot y = y \Longrightarrow$

$x \to y = \bar{x} \cdot \bar{y} \vee y$

3 Dualität der booleschen Funktionen und Formeln

3.1 Def. Sei $f : \{0,1\}^n \to \{0,1\}$. Dann definieren wir die duale Funktion $f^* : \{0,1\}^n \to \{0,1\}$ zu f durch

$$f^*(x_1, \ldots, x_n) := \overline{f(\overline{x_1}, \ldots, \overline{x_n})}.$$

$$(f^*)^* = \overline{(\overline{f(\overline{x_1}, \ldots, \overline{x_n})})^*}$$

$$= \overline{\overline{f(\overline{x_1}, \ldots, \overline{x_n})}}$$

$$= f(x_1, \ldots, x_n) = f$$

3.2 Def. Als duale Formel zu einer booleschen Formel F definieren wir die Formel F^* die aus F entsteht, indem in der Formel die Operatioen $\wedge$ und $\vee$ sowie die Konstanten 0 und 1 vertauscht. Mit anderen Worten ensteht die duale Formeln nach durch die folgenden Regeln:

- $L^* := L$ für jedes Literal L,
- $0^* = 1$ und $1^* = 0$
- $(A \vee B)^* = A^* \wedge B^*$ und $(A \wedge B)^* = A^* \vee B^*$.
- $(\overline{A})^* = \overline{A}$

$(x_i)^* = \overline{\overline{x_i}} = x_i$

$(\overline{x_i})^* = \overline{\overline{x_i}} = \overline{x_i}$

$\Rightarrow L^* = L$ für jedes Literal.

3.3 Thm. *Sei F boolesche Formel. Dann ist die duale Funktion zur Funktion von F gleich der Funktion zur dualen Formel F^*.*

Beweis. Die Behauptung folgt aus der Tatsache, dass für die Operation $f \mapsto f^*$ für boolesche Funktionen $f, g : \{0,1\}^n \to \{0,1\}$ die folgenden Regeln gelten:

$$0^* = 1$$
$$1^* = 0$$
$$(x_i)^* = x_i$$
$$(\neg f)^* = \neg f^*$$
$$(f \vee g)^* = f^* \wedge g^*$$
$$(f \wedge g)^* = f^* \vee g^*$$

Diese Regeln entsprechen genau den Regeln, die man benutzt, wenn man eine boolesche Formel

$$\left(\overline{f(x_1, \ldots, x_n)}\right)^* = \overline{f(\bar{x}_1, \ldots, \bar{x}_n)} = f(\bar{x}_1, \ldots, \bar{x}_n)$$

$$\overline{\left(f(x_1, \ldots, x_n)^*\right)} = \overline{f(\bar{x}_1, \ldots, \bar{x}_n)} = f(\bar{x}_1, \ldots, \bar{x}_n)$$

$$\left(f(x_1, \ldots, x_n) \vee g(x_1, \ldots, x_n)\right)^* = \overline{f(\bar{x}_1, \ldots, \bar{x}_n) \vee g(\bar{x}_1, \ldots, \bar{x}_n)}$$

$$= \overline{f(\bar{x}_1, \ldots, \bar{x}_n)} \wedge \overline{g(\bar{x}_1, \ldots, \bar{x}_n)}$$

$$= f^* \wedge g^*$$

De Morgan

$$\overline{A \vee B} = \bar{A} \wedge \bar{B}$$

dualisiert. Das bedeutet, dass man die Behauptung des Theorems etwa durch Induktion nach der Anzahl der Operationen $\neg, \wedge, \vee$, die man in F verwendet, beweisen kann. $\square$

De Morgan.

$$\exists \iff \bigvee$$

$$\forall \iff \bigwedge$$

$$\exists\, x \in \mathbb{R} : P(x) \qquad \bigvee_{x \in \mathbb{R}} P(x)$$

$$\exists\, i \in \{1,2,3\} : P(i) \quad = \quad P(1) \vee P(2) \vee P(3) \quad = \quad \bigvee_{i=1}^{3} P(i)$$

$$\forall x \in \mathbb{R} : P(x) \qquad \bigwedge_{x \in \mathbb{R}} P(x)$$

$$\forall i \in \{1,2,3\} : P(i) \quad = \quad P(1) \wedge P(2) \wedge P(3) \quad = \quad \bigwedge_{i=1}^{3} P(i)$$

De Morgan:

$$\overline{A \vee B} = \overline{A} \wedge \overline{B} \qquad (A, B - \text{2 Stück})$$

$$\overline{\bigvee_{i=1}^{n} A_i} = \bigwedge_{i=1}^{n} \overline{A_i} \qquad (A_1, \ldots, A_n - \text{n Stück})$$

$$\overline{\bigvee_{x \in X} A(x)} = \bigwedge_{x \in X} \overline{A(x)} \qquad (X \text{ kann beliebig groß sein})$$

$$\overline{\exists x \in X : A(x)} = \forall x \in X : \overline{A(x)}$$

Bsp.

$$(x \leftrightarrow y)^* = (x \cdot y \vee \bar{x} \cdot \bar{y})^*$$

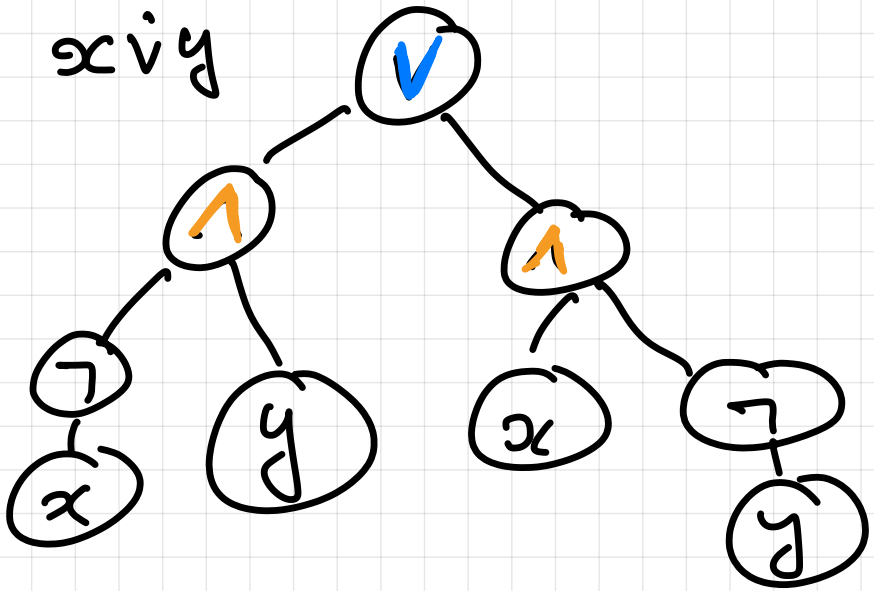
$$= (x \cdot y)^* \cdot (\bar{x} \cdot \bar{y})^*$$

$$= (x \vee y) \cdot (\bar{x} \vee \bar{y})$$

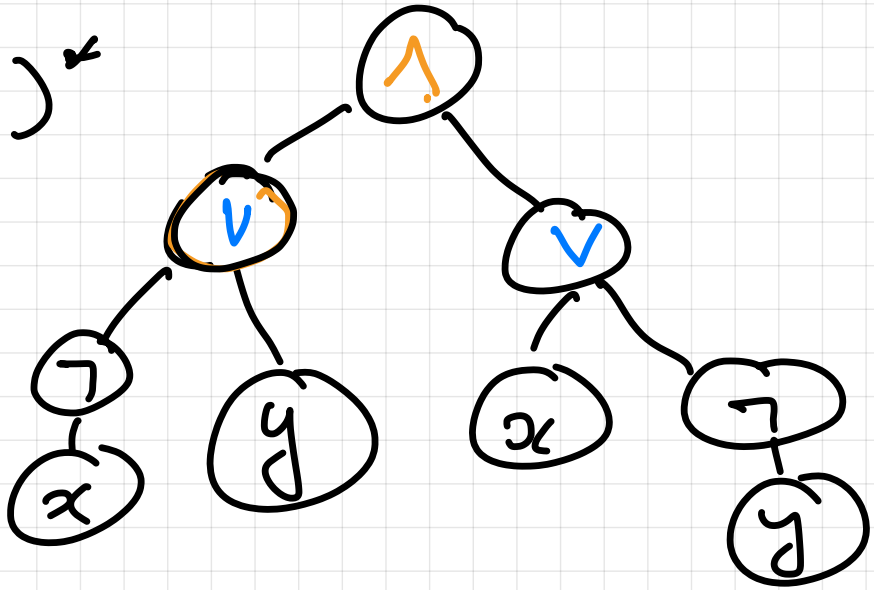
$$(x \dot{\vee} y)^* = (\bar{x} \cdot y \vee x \cdot \bar{y})^*$$

$$= (\bar{x} \vee y) \cdot (x \vee \bar{y})$$

$x \dot{\vee} y$



$(x \dot{\vee} y)^*$



x	y	$f = x \dot{\vee} y$	f^*	$(\bar{x} \vee y) \cdot (x \vee \bar{y})$
0	0	0	1	1
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

$$(x \dot{\vee} y)^* = x \leftrightarrow y$$

$$f(x,y)^* = \overline{f(x,y)}$$

$$f^*(0,0) = \overline{f(0,0)} = \overline{f(1,1)}$$

Bsp.

x_1	x_2	x_3	f	f^*
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	0	

4 Darstellung von booleschen Formeln durch KNFs

4.1 Thm. *Jede boolesche Funktion* $f : \{0,1\}^n \to \{0,1\}$ *kann durch eine KNF gegeben werden.*

Beweis. Wir starten mit der Darstellung von f^* als

$$f^*(x_1, \ldots, x_n) = \bigvee_{f^*(a_1,\ldots,a_n)=1} \bigwedge_{i=1}^{n} (x_i \leftrightarrow a_i).$$

$x_i \leftrightarrow 1 = x_i$
$x_i \leftrightarrow 0 = \overline{x_i}$

$\overline{x_i \leftrightarrow a_i} = \overline{x_i} \leftrightarrow a_i$

Das ergibt

$$\overline{f(\overline{x_1}, \ldots, \overline{x_n})} = \bigvee_{f(\overline{a_1},\ldots,\overline{a_n})=0} \bigwedge_{i=1}^{n} (x_i \leftrightarrow a_i).$$

Wir negieren die Formel und erhalten mit Hilfe des Gesetzes von De Morgan die Darstellung

$$f(\overline{x_1}, \ldots, \overline{x_n}) = \bigwedge_{f(\overline{a_1},\ldots,\overline{a_n})=0} \bigvee_{i=1}^{n} (\overline{x_i} \leftrightarrow a_i).$$

Substitution von x_i durch $\overline{x_i}$ ergibt

$$f(x_1, \ldots, x_n) = \bigwedge_{f(\overline{a_1},\ldots,\overline{a_n})=0} \bigvee_{i=1}^{n} (x_i \leftrightarrow a_i)$$

$a_i \in \{0,1\}$

$b_i = \overline{a_i} \in \{0,1\}$

$a_i = b_i$

Substitution von a_i durch $\overline{b_i}$ ergibt

$$f(x_1, \ldots, x_n) = \bigwedge_{f(b_1,\ldots,b_n)=0} \bigvee_{i=1}^{n} (x_i \leftrightarrow \overline{b_i}) \tag{V.4}$$

Wegen (V.1) kann $x_i \leftrightarrow \overline{b_i}$ je nach dem Wert von b_i entweder durch x_i oder durch $\overline{x_i}$ ersetzt werden. Mit dieser Interpretation ist die rechte Seite von (V.4) eine KNF. □

Bsp.

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$$f = (x_1 \vee x_2 \vee x_3) \wedge$$
$$(x_1 \vee \bar{x}_2 \vee x_3) \wedge$$
$$(\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge$$
$$(\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge$$
$$(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

Bsp.

x	y	$x \leftrightarrow y$	$x \dot{\vee} y$	$x \rightarrow y$
0	0	1	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	0	1

$$x \leftrightarrow y = (x \vee \bar{y}) \wedge (\bar{x} \vee y)$$

$$x \dot{\vee} y = (x \vee y) \wedge (\bar{x} \vee \bar{y})$$

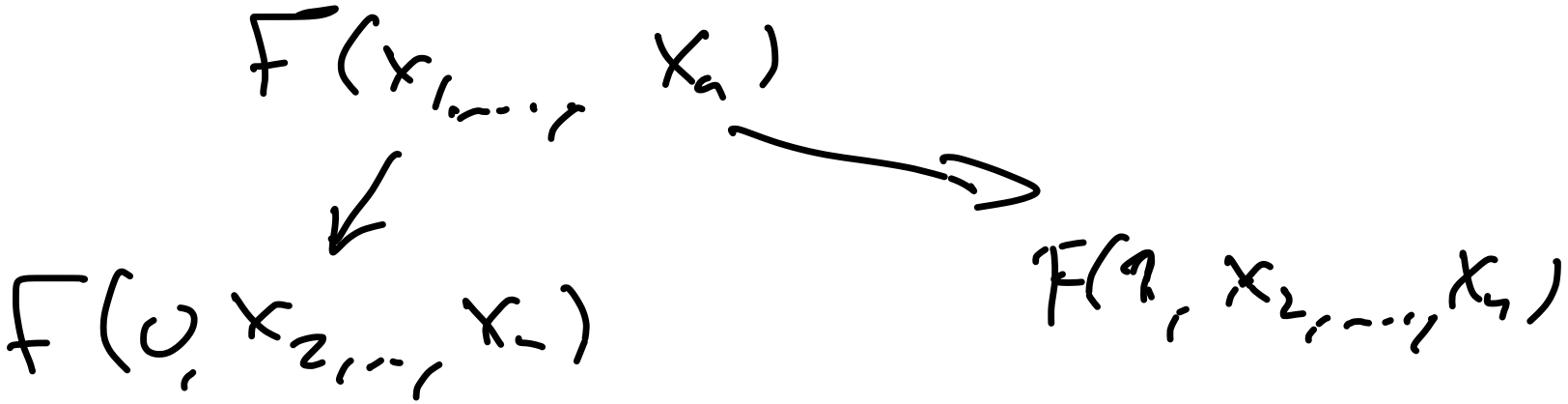
$$x \rightarrow y = \overline{x} \vee y$$

4.2. (V.4) ist ein Rezept zur Konstruktion einer Darstellung als KNF für ein gegebenes f. Für jede Belegung $(b_1, \ldots, b_n)$, bei der f falsch wird, füge die folgende Elementardijsunktion hinzu, deren Literarale für jedes $i = 1, \ldots, n$, folgendermaßen fixiert werden: x_i bei $b_i = 0$ und $\overline{x_i}$ bei $b_i = 1$. Dieses Rezept ist die "Dualisierung" des Rezepts (2.4) zur Konstruktion, das dadurch entsteht, dass man die Rollen von 0 und 1 vertauscht.

5 Erfüllbarkeitsproblem der Aussagenlogik

5.1 Def. Als SAT bezeichent man das Rechenproblem, bei dem man für eine gegebene KNF F entscheiden muss, ob F erfüllbar ist. Als Sprache ist SAT wie folgt gegeben:

$$\mathrm{SAT} = \{\llcorner F \lrcorner : F \text{ erfüllbare KNF}\}$$



5.2. Es ist klar, dass SAT entscheidbar ist, weil man bei einer KNF F mit n Variablen, alle 2^n Belegungen durchprobieren kann, um zu testen, ob F erfüllbar ist. Dieser Ansatz zur Entscheidung von SAT ergibt allerdings kein Polynomialzeit-Algorithmus.

2^{100}

5.3 (Eine der wichtigsten Vermutungen in der Mathematik bzw. Theoretischen Informatik)**.** Es wird vermutet, dass SAT nicht in Polynomialzeit entscheidbar ist. Dieses Problem ist insofern sehr wichtig, als in den Anwendungen buchstäblich Tausende von Problemen bekannt sind, die zu SAT im Sinne der Polynomialzeit-Karp-Reduktion äquivalent sind. Es handelt sich um die sogenannten NP-vollständigen Probleme. Mehr dazu erfahren Sie in den weiterführenden Kursen.

Bsp. Reduktion von Färbung zu SAT.

Gegeben: Graph $G = (V, E)$ und die Anzahl der Farben k.

$v \in V$ mit $x_{v,1}, \ldots, x_{v,k}$ Boolesche Variablen.

$x_{v,i}$: = "Der Knoten v wurde mit der Farbe i gefärbt".

Insgesamt: $|V| \cdot k$ Boolesche Variablen.

Die Variablen sollen zulässig belegt werden.

Bedingungen: $1 \le i < j \le k$, $v \in V$

$\overline{x_{v,i}} \vee \overline{x_{v,j}}$

(eine der beiden Farben wird nicht für v benutzt)

$\bigvee_{i=1}^{n} x_{v,i}$

(eine der k Farben wird benutzt).

$1 \le i \le k$, $\{u,v\} \in E$:

$\overline{x_{u,i}} \vee \overline{x_{v,i}}$

(Farbe i wird für u oder für v nicht benutzt.)

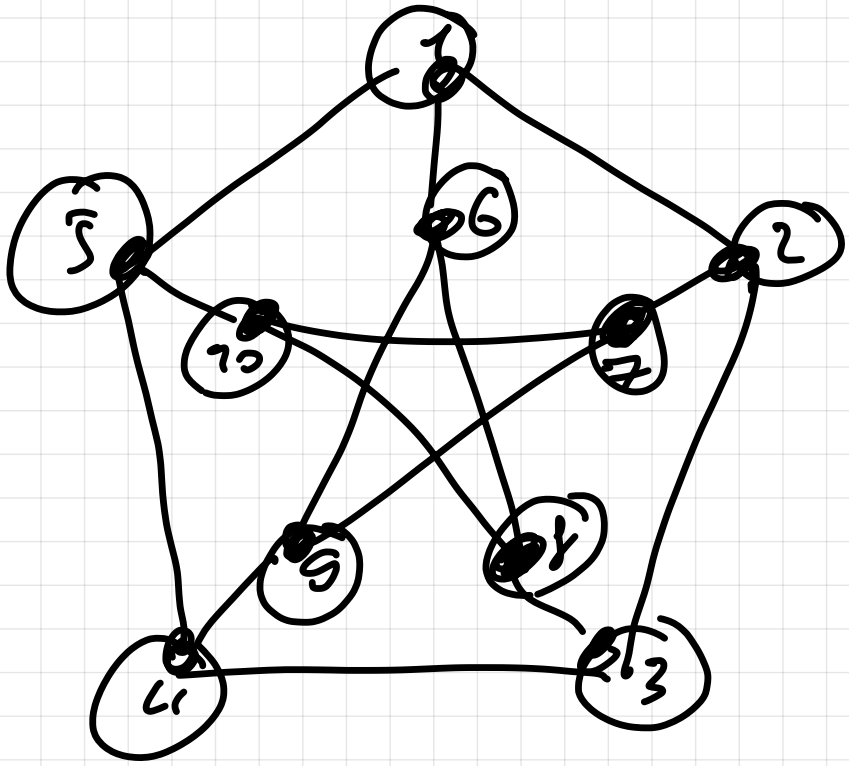
$G = (V, K)$, k ⟶ Polynomial-Karp-reduction.

KNF:

$$\bigwedge_{1 \le i < j \le k} \bigwedge_{v \in V} \left(\overline{x_{v,i}} \vee \overline{x_{v,j}} \right) \wedge \bigwedge_{v \in V} \left(\bigvee_{i=1}^{k} x_{v,i} \right) \wedge$$

$$\cdot \bigwedge_{1 \le i \le k} \bigwedge_{\{u,v\} \in E} \left(\overline{x_{u,i}} \vee \overline{x_{v,i}} \right)$$

Noch knobeln:



$2 \cdot 10 + 15 \cdot 2$

$= 50$ Klauseln.

20 Variablen.

R, B

Rot, Blau

$r_i, b_i \in \{0,1\}$

$$\bigwedge_{i=1}^{10} (\overline{r_i} \vee \overline{b_i}) \wedge (r_i \vee b_i)$$

$$\wedge (\overline{r_1} \vee \overline{r_2}) \wedge (\overline{b_1} \vee \overline{b_2}) \wedge$$

$$\wedge (\overline{r_2} \vee \overline{r_3}) \wedge (\overline{b_2} \vee \overline{b_3}) \wedge$$

$$\vdots$$

$$\wedge (\overline{r_8} \vee \overline{r_{10}}) \wedge (\overline{b_8} \vee \overline{b_{10}})$$

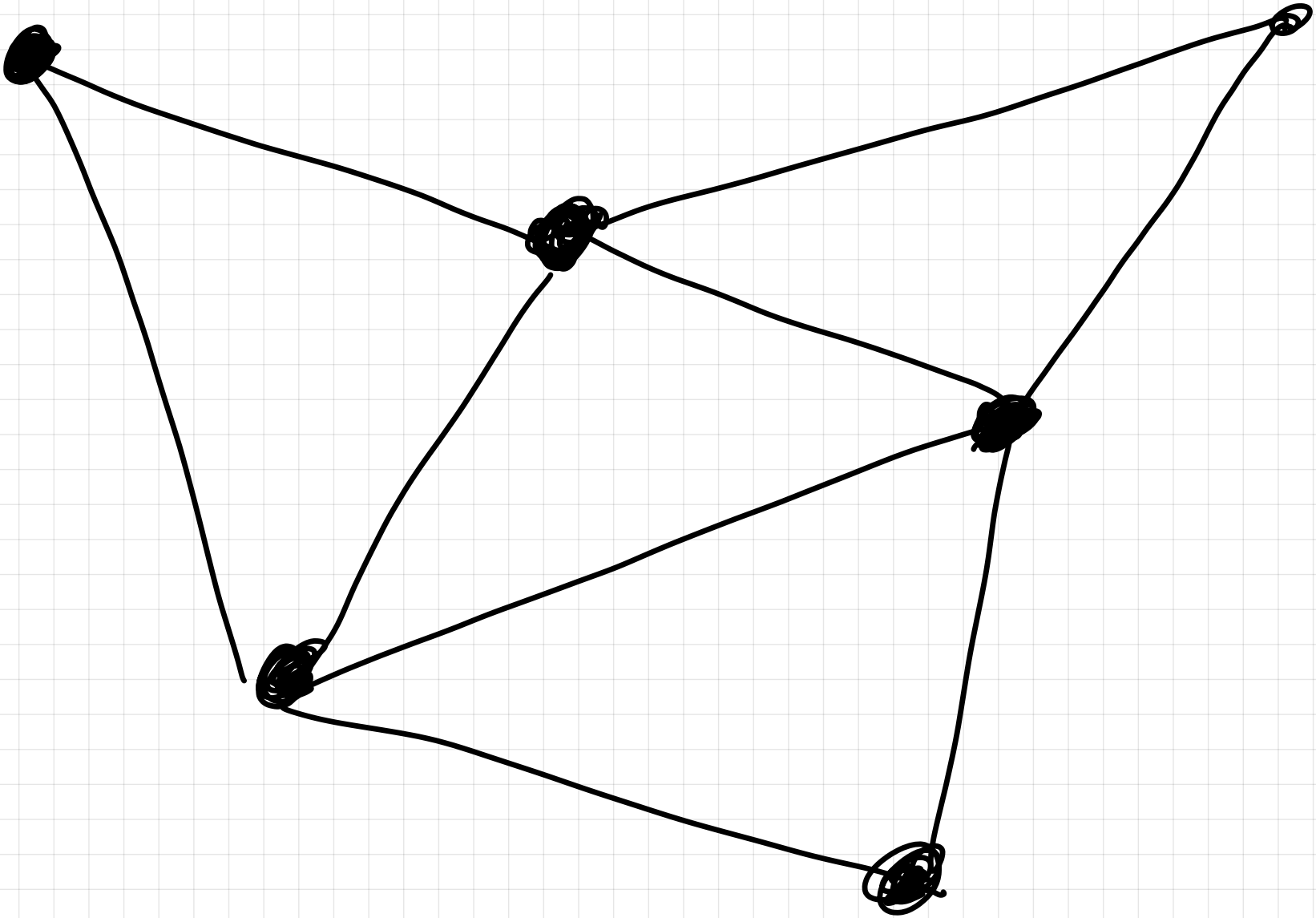
Ganzzahlige lineare Unfeasibility

$$\left.\begin{aligned} r_i + b_i &\geq 1 \\ r_i + b_i &\leq 1 \end{aligned}\right\} \Longleftrightarrow r_i + b_i = 1$$

$$r_1 + r_2 \leq 1$$
$$b_1 + b_2 \leq 1$$
$$\vdots$$
$$r_8 + r_{10} \leq 1$$
$$b_8 + b_{10} \leq 1$$

$$0 \leq r_i \leq 1$$
$$0 \leq b_i \leq 1$$

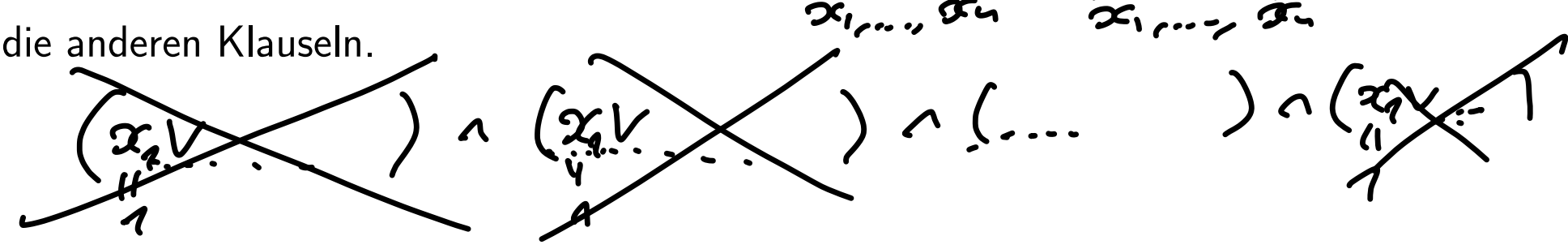
$$r_i, b_i \in \mathbb{Z}$$



5.4. Der DPLL Algorithmus zur Berechnung ist ein rekursiver Algorithmus zur Entscheidung von SAT, in dem die folgenden zwei Techniken die Laufzeit etwas verbessern sollen:

Unit propagation. Bei einer Klausel aus genau einem Literal a kommt für die Erfüllung der Klausel nur die Zuweisung $a = 1$ in Frage. Wenn man den Wert $a = 1$ für a und somit auch den Wert $\overline{a} = 0$ für $\overline{a}$ zugewiesen hat, so kann man die Klauseln mit a weglassen. Bei den Klauseln mit dem Literal $\overline{a}$, kann dieses Literal gestrichen werden, sodass potenziell weitere Klauseln entstehen, die nur ein Literal haben. Man kann also den Prozess der Zuweisung solange wiederholen, bis man keine Klauseln aus genau einem Literal hat.

Pure Literal Assignment. Kommt ein Literal a in den Klauseln vor, ohne dass $\overline{a}$ in irgendeiner Klausel vorkommt, so kann die Zuweisung $a = 1$ für das Literal a festgelegt werden. Diese Zuweisung erfüllt die Klauseln mit dem Literal a und hat keinen Einfluss auf die anderen Klauseln.



Sobald die zugrundeliegende KNF nicht durch die beiden beschriebenen Operationen vereinfacht werden kann, wählt man eine Variable x_i und probiert zur Erfüllung der Formel die beiden Möglichen Zuweisungen $x_i = 0$ sowie $x_i = 1$ mit Hilfe von rekursiven Aufrufen von DPLL aus. Hat der erste der beiden rekursiven Aufrufe die Erfüllbarkeit festgestellt, so muss man den zweiten Aufruf nicht mehr starten. Auf diese Weise kann für manche Eingabeinstanzen eine vollständige Aufzählung aller möglichen Belegungen der Variablen vermieden werden.

5.5 Lem (Resolution)**.** Die Formel $(x \vee y)(\overline{x} \vee z) \to (y \vee z)$ ist Tautologie.

Beweis. Das lässt sich durch die Tafel für diese Formel verifizieren. Alternativ kann man es auch so sehen: Setzen wir $x = 0$ ein, so reduziert sich die Formel zur Tautologie $y \to (y \vee z)$ (mit Worten: aus y folgt y oder z). Setzen wir $x = 1$ ein, so reduziert sich die Formel zur Tautologie $z \to (y \vee z)$ (mit Worten: aus z folgt y oder z). □

$(a \to x) \wedge (x \to b) \to (a \to b) =$

$(\bar{a} \vee x) \wedge (\bar{x} \vee b) \to (\bar{a} \vee b)$.

$\bar{a} = y$ $b = z$

5.6 Def. Für eine Variable x und zwei Klauseln G und H der Form $G = A \vee x$ und $H := B \vee \overline{x}$, nennt man die Klausel $A \vee B$ die **Resolvente** von G und H bzgl. der Variablen x. Herleitung einer Resolvente aus zwei Klauseln nennt man **Resolution**.

Wir sagen, dass eine Klausel K aus einer KNF $F = \bigwedge_{i=1}^{m} K_i$ mit Klauseln $K_1, \ldots, K_m$ durch einen **Resolventenbeweis** herlgeleitet werden kann, wenn K eine der Klauseln $K_1, \ldots, K_m$ ist oder eine endliche Folge $K_{m+1}, \ldots, K_t$ von Klauseln existiert derart, dass für jedes $i = m+1, \ldots, t$, die Klausel K_i Resolvente von zwei Klauseln aus $K_1, \ldots, K_{i-1}$ ist und die letzte Klausel K_t dieser Folge gleich K ist. Hierbei nennt man $K_{m+1}, \ldots, K_{m+t}$ die Beweisschritte und t die Länge des Beweises. Ist K direkt in der KNF enthalten, so hat K einen Beweis der Länge 0.

Wir sagen, dass eine KNF F mit Klauseln $K_1, \ldots, K_m$ durch einen **Resolutionsbeweis widerlegt** werden kann, wenn die Klausel 0 aus F durch einen Resolventenbeweis hergeleitet

werden kann.

5.7 Thm. *Eine KNF F ist genau dann nicht erfüllbar, wenn sie durch einen Resolutionsbeweis widerlegt werden kann.*

Beweis. Aus Lemma 5.5 folgt: ist F eine KNF und R Resolvente von zwei Klauseln aus F, so ist $F \wedge R$ äquivalent zu F. Also ist jede KNF, die durch einen Resolutionsbeweis widerlegt werden kann, nicht erfüllbar, weil die Klausel 0, die durch den Resolutionsbeweis erzeugt wird, nicht erfüllbar ist.

Wir zeigen, dass jede nicht-erfüllbare KNF F durch einen Resolutionsbeweis widerlegt werden kann. Wir führen dabei die Induktion über die Anzahl $n \in \mathbb{N}_0$ der Variablen $x_1, \ldots, x_n$ in der KNF F. Ist $n = 0$, so ist $F = 0$ durch einen Resolutionsbeweis der Länge 0 widerlegt.

Sei $n \in \mathbb{N}$ so, dass jede nicht-erfüllbare KNF in höchstens $n - 1$ Variablen durch Resolutionsbweis widerlegt werden kann.

Wir teilen die Klauseln von F in drei Arten auf, je nachdem, ob die Klausel x_n oder $\overline{x_n}$ oder

weder x_n noch $\overline{x_n}$ als einen der Literale enthält. Dem entsprechend schreiben wir die KNF F in n Variablen als $F = A \wedge B \wedge C$, mit

$$A = \bigwedge_{i \in I} A_i \qquad B = \bigwedge_{j \in J} (B_j \vee x_n) \qquad C = \bigwedge_{k \in K} (C_k \vee \overline{x_n})$$

mit den Klauseln A_i, B_j, C_k die nur von $x_1, \ldots, x_{n-1}$ abhängig sind. Ist J die leere Indexmenge, so ist $B = 1$ und wir können C durch die Wahl $x_n = 0$ erfüllen. Somit ist die Erfüllbarkeit von F in diesem Fall äquivalent zur Erfüllbarkeit von A. Die Behauptung folgt dann aus der Induktionsvoraussetzung. Ist K die leere Menge, so ist $C = 1$ und wir können B durch die Wahl $x_n = 1$ erfüllen. Somit ist die Erfüllbarkeit von F in diesem Fall ebenfalls äquivalent zur Erfüllbarkeit von A, sodass man ebenfalls die Induktionsvoraussetzung benutzt kann.

Sind J und K beide nicht leer, so stellt sich heraus, dass die Erfüllbarkeit von F zur

Erfüllbarkeit der Formel

$$G := A \wedge \bigwedge_{\substack{j \in J \\ k \in K}} (B_j \vee C_k). \tag{V.5}$$

äquivalent ist. Um das zu sehen, bemerken wir zuerst, dass $B_j \vee C_k$ die Resolvente von $B_j \cap x_n$ und $C_k \cap \overline{x_n}$ bzgl. x_n ist. Aus der Erfüllbarkeit von F folgt also die Erfüllbarkeit von G. Umgekehrt, sei G erfüllbar. Die Formel G kann also

$$G = A \wedge \left(\left(\bigwedge_{j \in J} B_j \right) \vee \left(\bigwedge_{k \in K} C_k \right) \right)$$

umformuliert werden. Gibt es eine Belegung $b = (b_1, \ldots, b_{n-1}) \in \{0,1\}^n$ der Variablen $x_1, \ldots, x_{n-1}$ mit $G(b) = 1$, so gilt $A(b) = 1$ und mindestens einer der folgenden Bedingungen ist erfüllt: $B_j(b) = 1$ für alle $j \in J$ oder $C_k(b) = 1$ für alle $k \in K$. Im ersten Fall ist $F(b_1, \ldots, b_{n-1}, 0) = 1$ und im zweiten Fall ist $F(b_1, \ldots, b_{n-1}, 1) = 0$.

Esp.

$$\bigwedge_{\substack{j=1,2\\k=1,2}} (B_j \vee C_k)$$

$$= \underbrace{(B_1 \vee C_1) \wedge (B_1 \vee C_2)} \wedge \underbrace{(B_2 \vee C_1) \wedge (B_2 \vee C_2)}$$

$$= \left(B_1 \vee \underbrace{(C_1 \wedge C_2)}\right) \wedge \left(B_2 \wedge \underbrace{(C_1 \vee C_2)}\right).$$

$$= (B_1 \wedge B_2) \vee (C_1 \vee C_2)$$

$$= \left(\bigwedge_{j=1,2} B_j\right) \vee \left(\bigwedge_{k=1,2} C_k\right).$$

Distributivgesetze:

$\mathbb{R}$

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$a + (b \cdot c) \neq (a + b) \cdot (a + c)$$

$\{0,1\}$ Boolesche Algebra.

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

Ist F nicht erfüllbar, dann ist also auch G nicht erfüllbar. Nach der Induktionsvoraussetzung kann G durch einen Resolutionsbeweis widerlegt werden. Jede Klausel in (V.5) ist aber eine Klausel on F oder eine Resolvente von zwei Klauseln aus F. Somit lässt sich der Resolutionsbweis der Nicht-Erfüllbarkeit von G zu einem Resolutionsbeweis der Nicht-Erfüllbarkeit von F ergänzt werden. □

5.8 Def. Für $m \in \mathbb{N}$ nennen wir eine KNF, bei der jede Klausel höchstens m Literale enthält eine m-KNF.

Als m-SAT bezeichnen wir das Entscheidungsproblem, bei dem man für eine gegebene m-KNF testet, ob diese erfüllbar ist. Als Sprache ist m-SAT folgendermaßen definiert:

$$m\text{-SAT} = \{\llcorner F \lrcorner : F \textit{ erfüllbare } m\textit{-KNF}\}$$

5.9 Thm. 2-SAT *ist Polynomialzeit-entscheidbar.*

Beweis. Sei F eine 2-KNF mit Variablen $x_1, \ldots, x_n$. Ohne Beschränkung der Allgemeinheit kann vorausgesetzt werden, dass jede Klausel genau zwei Literale enthält.

Wir führen einen Digraphen $D = (V, A)$ ein mit $V = \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$ und mit der Kantenmenge

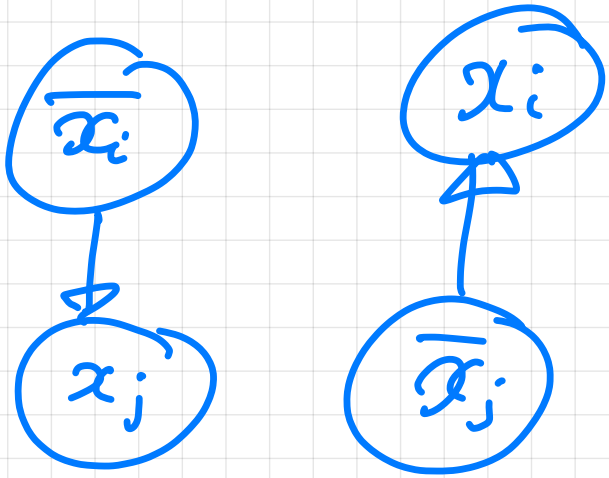
$$A = \{(a, b) \,:\, \{\overline{a}, b\} \text{ ist die Menge der Literale einer der Klauseln von } F\}$$

Mit anderen Worten stellt jede Kante (a, b) die Implikation $a \to b$ dar, welche in der Formel eine Klausel $\overline{a} \vee b$ in F vorhanden ist. Die bekannte Tautologie

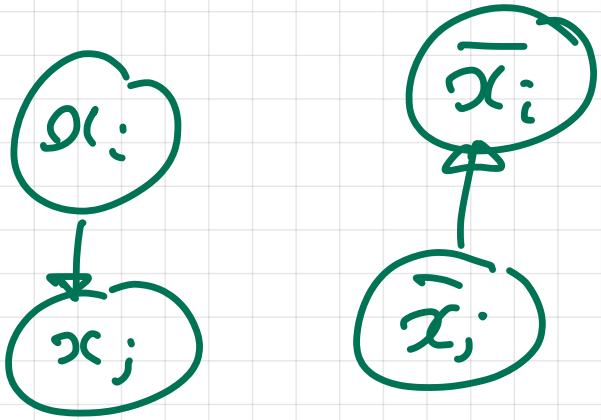
$$(a \to b)(b \to c) \to (a \to c)$$

ist nichts anderes als die Resolution $(b \vee \overline{a})(\overline{b} \vee c) \to (\overline{a} \vee c)$, mit der man aus den Klauseln $b \vee \overline{a}$ und $\overline{b} \vee c$ die Klausel $\overline{a} \vee c$ erhält. Ein (u, v)-Pfad in D entspricht also der Herleitung von

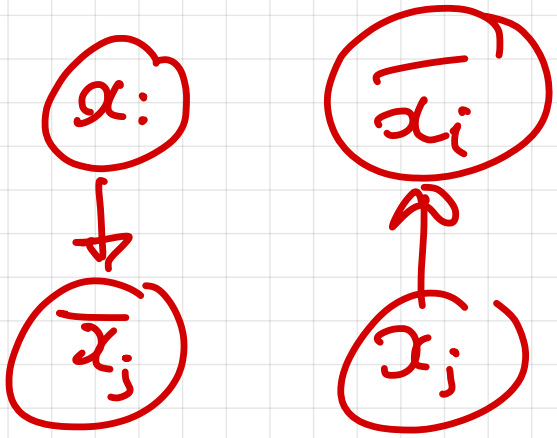
$x_i \vee x_j = \bar{x}_i \to x_j = \bar{x}_j \to x_i$

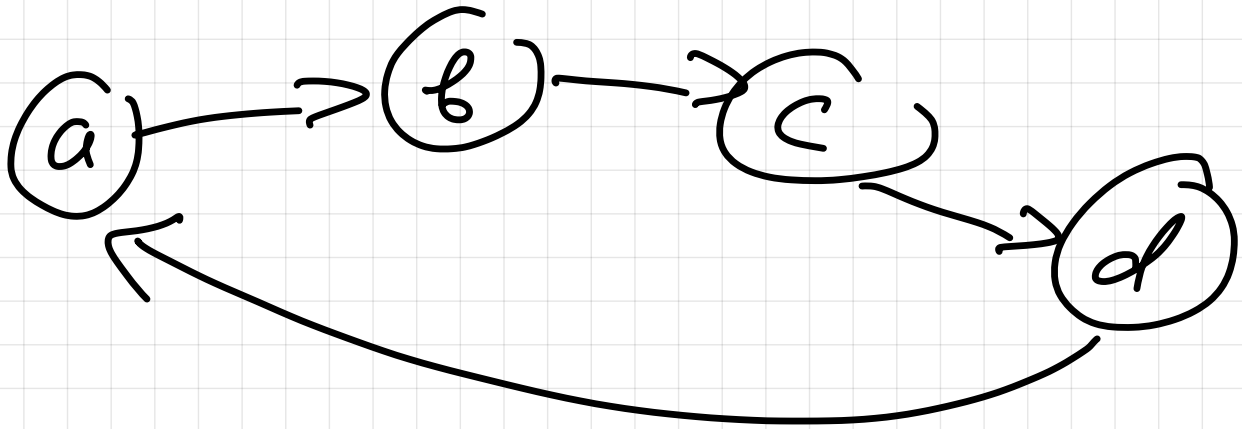


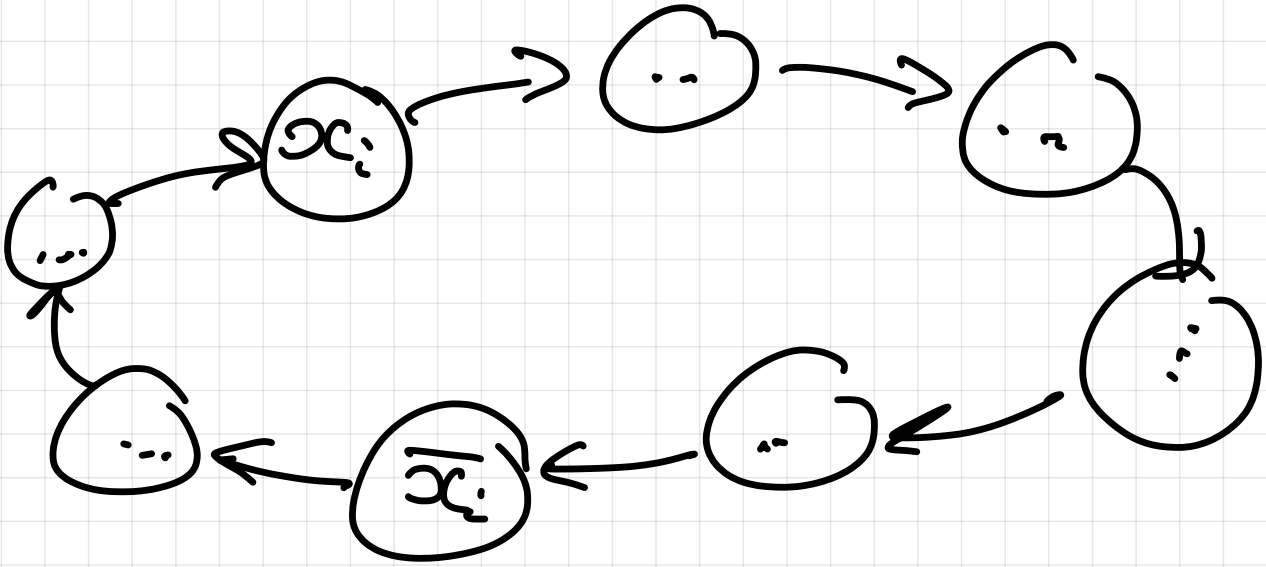
$\bar{x}_i \vee x_j = x_i \to x_j = \bar{x}_j \to \bar{x}_i$



$\bar{x}_i \vee \bar{x}_j = x_i \to \bar{x}_j = x_j \to \bar{x}_i$







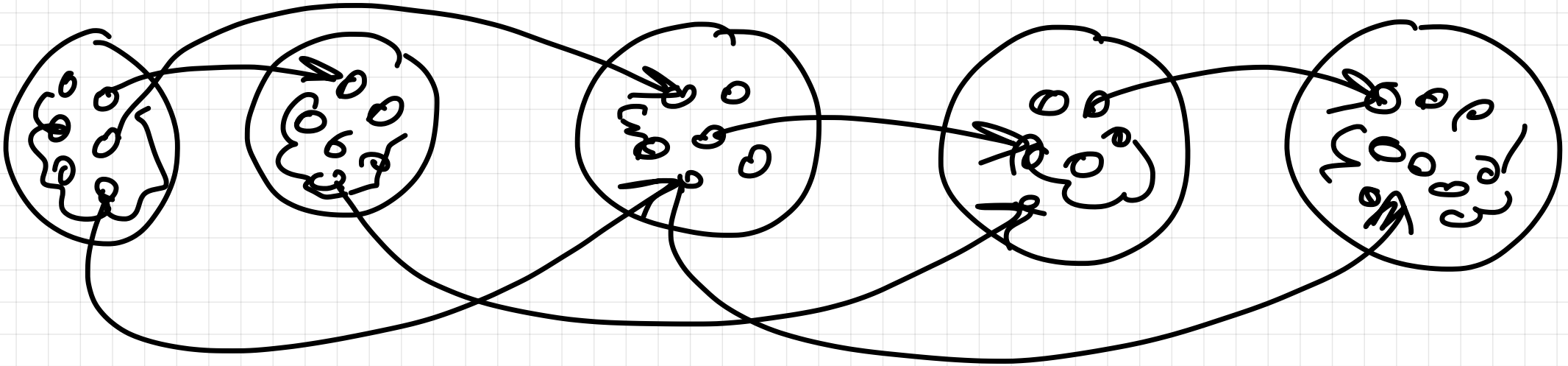
$\Rightarrow$ (u,v) - Pfad in G $\Leftrightarrow$ $(u \to v)$ folgt aus den Klauseln

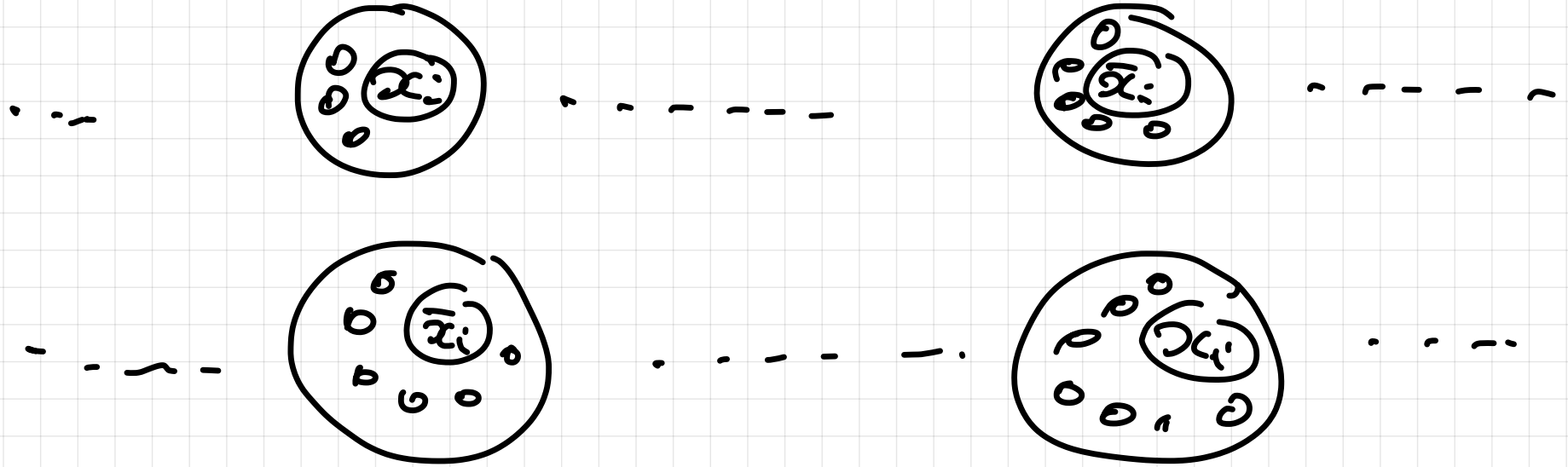
$u \to v = \overline{u} \vee v$ durch einen Resolutionsbeweis. Wenn für ein $i = 1, \ldots, n$ die Knoten x_i und $\overline{x_i}$ in D gegenseitig erreichbar sind, so können wir $\overline{x_i} \to x_i = x_i$ sowie $x_i \to \overline{x_i} = \overline{x_i}$ mit Hilfe der Resolutionsbweise herleiten. Da man aber x_i und $\overline{x_i}$ nicht gleichzeitig erfüllen kann, folgt in diesem Fall, dass F nicht erfüllbar ist. Wir stellen also fest, dass F nicht erfüllbar ist, wenn für ein $i = 1, \ldots, n$ die beiden Atome x_i und $\overline{x_i}$ in der selben starken Zusammenhangskomponente von D liegen.

Da man die starken Zusammenhangskomponenten mit Hilfe einer Tiefensuche berechnen kann, lässt sich der beschriebene Fall in Polynomialzeit abfangen.

Andernfalls können wir eine Belegung wie folgt festlegen. Wir sortieren den Komponentengraphen von D topologisch (das geht ebenfalls mit der Tiefensuche) und legen $x_i = 0$ genau dann fest, wenn die Komponenten von x_i vor der Komponente $\overline{x_i}$ aufgelistet wird. Da wir die Klauseln als Kanten (a, b) mit $a, b \in V$ kodieren und dabei als Implikationen $a \to b$ umformulieren, bleibt es zu zeigen, dass jede solche Implikation durch unsere Belegung erfüllt wird.

G $\rightsquigarrow$ Graph der Starken Zusammenhangskomponenten





Wäre $a \to b$ nicht erfüllt, so würde es bedeuten, dass man die Werte $a = 1$ und $b = 0$ zugewiesen hat. Das bedeutet wiederum, dass die Komponente von $\overline{a}$ vor der Komponente von a ist und die Komponente von b vor der Komponenten von $\overline{b}$ aufgelistet ist. Wegen $a \to b$ ist die Komponente von a vor der Komponente von b aufgelistet. Somit ist die Komponente von $\overline{a}$ vor der Komponente von $\overline{b}$ aufgelistet. Nach der Konstruktion von $D = (V, A)$ ist in A neben der Kante (a, b) auch die Kante $(\overline{b}, \overline{a})$ vorhanden. Diese Kante führt von der Komponente von $\overline{b}$ zur Komponente von $\overline{a}$, was der Tatsache widerspricht, dass unsere Auflistung der Komponenten eine topologische Sortierung ist. Dies zeigt, dass jede Klausel von F durch die von uns eingeführte Belegung erfüllt wird. □

$\overline{a} \vee b = a \longrightarrow b.$

$a \in \{x_i, \overline{x}_i\}$

$b \in \{x_j, \overline{x}_j\}$

so eine kann es nicht geben, denn

der Graph der starken Zusammenhangs-

komponenten ist topologisch sortiert

$a = x_i$ $\quad b = x_j$

$\overline{x_i} \vee x_j = x_i \rightarrow x_j = \overline{a_j} \rightarrow \overline{x_i}$

5.10 Def. Ist x Variable, so nennen wir x ein positives Literal und $\overline{x}$ ein negatives Literal. Eine Klausel mit höchstens einem positiven Literal nennen wir eine Hornklausel. Eine Klausel nennen wir eine Einheit, wenn sie nur aus einem Literal besteht.

Eine KNF aus Hornklauseln nennen wir eine Horn-KNF. Als $\mathrm{HORNSAT}$ bezeichnen wir das Problem, bei dem für eine gegebene Horn-KNF ihre Erfüllbarkeit entschieden wird. Als Sprache:

$$\mathrm{HORNSAT} := \{\llcorner F \lrcorner : F \text{ erfüllbare Horn-KNF}\}.$$

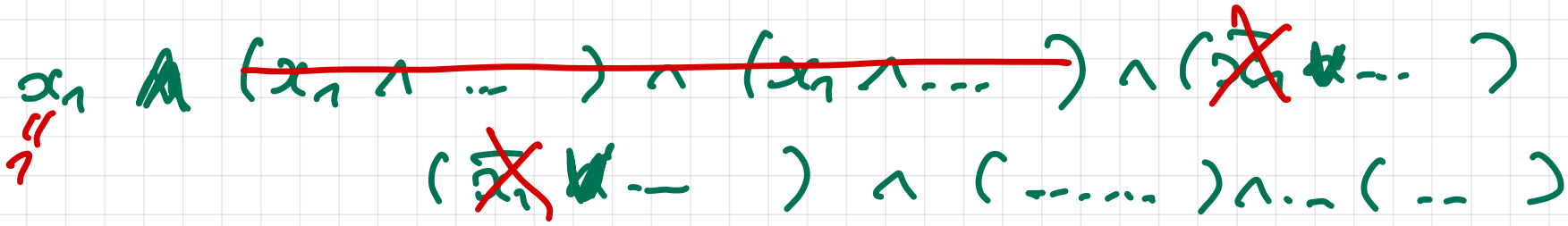
$\overline{x_i} \vee \overline{x_j} \vee x_k$

$\overline{x_i} \vee \overline{x_j} \vee \overline{x_k} \vee \overline{x_\ell}$

5.11 Thm. HORNSAT *ist Polynomialzeit-entscheidbar.*

Beweis. Wir benutzen die Unit Propagation, die bereits erwähnt wurde. Sei F eine Horn-KNF. Der Algorithmus, modifiziert F iterativ, ohne die Erfüllbarkeit von F zu beeinträchtigen (eine erfüllbare F bleibt erfüllbar, eine unerfüllbare F bleibt unerfüllbar).

Hat F die Klausel 0, so ist F nicht erfüllbar. Ist eine der Klausel eine Einheit a, so müssen wir a mit 1 belegen, um F zu erfüllen. Auf diese Weise werden alle Klauseln, die a enthalten aus F entfernt, und aus den Klauseln, die $\overline{a}$ enthalten, wird das Literal $\overline{a}$ entfernt (man beachte, dass eine Klausel ohne Literal die Klausel 0 ist). Den vorigen Prozess wiederholt man iterativ solange, bis die Klausel 0 generiert wird oder man die Formel F ohne erzeugt. Im letzteren Fall lässt sich die Formel erfüllen, indem man alle übrig gebliebenen Variablen gleich 0 setzt. □



Jede Klausel ist eine Horn-Klausel

$$\overline{x_1} \vee \overline{x_2} \vee x_3$$

$$\overline{x_1} \vee x_2$$

$$\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee x_{10}$$

$x_i = 0 \quad \forall i$

Bsp.

$$F = (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge \bar{x}_5 \wedge (\bar{x}_3 \vee x_1) \wedge x_3 \wedge x_2 \wedge (\bar{x}_6 \vee x_4) \wedge x_6$$

$x_2 = 1$
$x_3 = 1$
$x_5 = 0$
$x_6 = 1$

$\bar{x}_3 \vee x_1 \rightsquigarrow x_1$

~~$\bar{x}_1 \vee \bar{x}_2 \vee x_3$~~

$\bar{x}_6 \vee x_4 \rightsquigarrow x_4$

$F \rightsquigarrow x_3 \vee x_4$

$x_3 = 1$
$x_4 = 1$

5.12 Thm. $3\text{-SAT} =_{\mathrm{P}} \mathrm{SAT}$.

Beweis. Die von 3-SAT zu SAT ist einfach die identische Abbildung. Die Reduktion von SAT zu 3-SAT erfolgt durch die Einführung von Zusatzvariablen (Details IM AUFBAU). □

Literaturverzeichnis

[AZ02] Aigner, Ziegler. Das Buch der Beweise. Springer 2002

[Ber17] Berghammer: Mathematik für Informatiker. Grundlegende Begriffe und Strukturen. Springer Vieweg 2017

[Ber19] Berghammer: Mathematik für Informatiker. Grundlegende Begriffe und Strukturen und ihre Anwendung. Springer Vieweg 2019

[Big05] Biggs. Discrete mathematics. Oxford University Press 2005

[Bri01] Mathematik für Inforamtiker. Einführung an praktischen Beispielen aus der Welt der Computer. München: Hanser 2001

[CLRS17] Cormen, T. H., Leiserson, C. E., Rivest, R., Stein, C. Algorithmen-Eine Einführung. De Gruyter Oldenbourg. 2017.

[GR14] Goebbels, Rethmann. Mathematik für Informatiker: eine aus der Informatik motivierte Einführung mit zahlreichen Anwendungs- und Programmbeispielen. Springer Vieweg 2014

[GS11] Heinz Peter Gumm, Manfred Sommer. Einführung in die Informatik, Oldenbourg Verlag 2011

[KK15] Knauer, Knauer. Diskrete und algebraische Strukturen - kurz gefasst. Springer Spektrum 2015.

[KP09] Kreußler, Pfister. Mathematik für Informatiker: Algebra, Analysis, Diskrete Strukturen. Springer 2009.

[LLM21] Lehman, Leighton, Meyer. Mathematics for Computer Science. Lecture notes at MIT. `https://courses.csail.mit.edu/6.042/spring18/mcs.pdf`

[Lov20] Lovász, László. Complexity of Algorithms. Lecture Notes. 2020. `https://web.cs.elte.hu/~kiraly/complexity.pdf`

[Sch12] Schubert. Mathematik für Informatiker: ausführlich erklärt mit vielen Programmbeispielen und Aufgaben.

[Ste01] Steger. Diskrete Strukturen 1. Kombinatorik, Graphentheorie, Algebra. Springer 2001

[Tit19] Peter Tittmann. Einführung in die Kombinatorik, 3. Auflage, Springer 2019