Einführung in die Informatik 2

– NP-Vollständigkeit –

Sven Kosub

AG Algorithmik/Theorie komplexer Systeme
Universität Konstanz

E 202 | Sven.Kosub@uni-konstanz.de | Sprechstunde: Freitag, 12:30-14:00 Uhr, o.n.V.

Sommersemester 2009

Polynomialzeit-Algorithmen

Intuition:

Berechnungsprobleme, die Polynomialzeit-Algorithmen erlauben, können als (theoretisch) effizient lösbar angesehen werden

Fragen:

- Gibt es Berechnungsprobleme, die nicht effizient lösbar sind?
- Wie können wir dies für ein konkretes Problem feststellen?
- Welche Auswege gibt es, um mit solchen Problemen umgehen zu können?
- ...

Eingabegrößen

uniforme Eingabegröße:

- Anzahl der Komponenten zur Beschreibung der Eingabe

- $|(x_1, \ldots, x_n)|_1 = n$ für ganze Zahlen x_i
- $|G|_1 = n + m$ für Graph $G = (V, E)$ mit $\|V\| = n$ und $\|E\| = m$
- $|s_0 s_1 \ldots s_{n-1}|_1 = n$ für Buchstaben s_i

logarithmische Eingabegröße:

- Anzahl der Bits zur Beschreibung von ganzen Zahlen

- $|n|_{\log} = 1 + \lfloor \log_2 n \rfloor$

Beachte:

- $|(2^n)_{10}|_{\log} = |(10^n)_2|_1 = n + 1$ (bei binärer Repräsentierung)
- $|\underbrace{1111\ldots1111}_{2^n\text{-mal}}|_1 = 2^n$ (bei unärer Repräsentierung)

Laufzeit

- Anzahl der elementaren Operationen eines Algorithmus (in Java)
- Notation: $t_A(x)$ ist Laufzeit des Algorithmus A auf Eingabe x
- worst-case: Laufzeit im schlechtesten Fall

$$t_A^{\max}(n) = \max\{t_A(x) \mid \ x \text{ ist Eingabe der Größe } |x| = n \ \}$$

- best-case: Laufzeit im besten Fall

$$t_A^{\min}(n) = \min\{t_A(x) \mid \ x \text{ ist Eingabe der Größe mit } |x| = n \ \}$$

- average-case: Laufzeit im Erwartungswert

$$t_A^{\text{avg}}(n) = \mathbf{E}_{|x|=n} t_A(x)$$

Beachte: Laufzeitaussagen hängen von Eingaberepräsentierung ab

Laufzeit

Fakten:

- $t_A^{\min}(n) \leq t_A^{\text{avg}}(n) \leq t_A^{\max}(n)$ für alle Eingabegrößen n

- $t_A^{\max}(n) \leq f(n) \iff t_A(x) \leq f(n)$ für alle Eingaben x mit $|x| = n$
- $t_A^{\max}(n) \geq g(n) \iff t_A(x) \geq g(n)$ für eine Eingabe x mit $|x| = n$

- $t_A^{\min}(n) \leq f(n) \iff t_A(x) \leq f(n)$ für eine Eingabe x mit $|x| = n$
- $t_A^{\min}(n) \geq g(n) \iff t_A(x) \geq g(n)$ für alle Eingaben x mit $|x| = n$

Laufzeit

typische obere (worst-case) Laufzeitschranken:

- konstant: $t_A^{\max}(n) \in O(1)$
- logarithmisch: $t_A^{\max}(n) \in O(\log n)$
- linear: $t_A^{\max}(n) \in O(n)$
- quadratisch: $t_A^{\max}(n) \in O(n^2)$
- polynomiell: $t_A^{\max}(n) \in O(n^k)$ für ein $k \in \mathbb{N}$
- exponentiell: $t_A^{\max}(n) \in 2^{O(n^k)}$ für ein $k \in \mathbb{N}$

wichtigste untere (worst-case) Laufzeitschranke

- superpolynomiell: $t_A^{\max}(n) \in \Omega(n^k)$ für alle $k \in \mathbb{N}$

Die Klasse P

Entscheidungsprobleme:

- versuchen nicht Funktionswerte zu berechnen, sondern nur ob Eingabe „gut“ oder „schlecht“ ist
- betrachten Teilmengen $L \subseteq M$ der Grundmenge M aller möglichen Eingaben
- charakteristische Funktion $c_L : M \to \{0, 1\}$ von L:

$$c_L(x) =_{\text{def}} \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{falls } x \notin L \end{cases}$$

- Algorithmus A entscheidet ein Problem L, falls A die Funktion c_L berechnet

Beachte: Funktionen f können als Entscheidungsprobleme kodiert werden ($L_f =_{\text{def}} \{(x, y) \mid f(x) \geq y\}$)

Die Klasse P

- Entscheidungsproblem L heißt polynomiell, falls es eine Algorithmus mit polynomieller Laufzeit gibt, der L entscheidet
- $\mathrm{P} =_{\mathrm{def}} \{L \mid L \text{ ist ein polynomielles Entscheidungsproblem}\}$
- P steht für die Klasse der effizient lösbaren Entscheidungsprobleme

Beispiele für Probleme in P:

- Menge der Palindrome, d.h. $L = \{w \mid w \in \{0,1\}^* \text{ und } w = w^R\}$
- Menge der matchenden Wortpaare, d.h.
  $$L = \{(s,t) \mid \ s \text{ kommt in } t \text{ als Teilwort vor}\}$$
- Menge der zusammenhängenden Graphen
- Menge der Graphen mit Euler-Kreis
- Menge der planaren Graphen
- Menge der Primzahlen, d.h. $L = \{x \mid x \in \mathbb{N} \text{ und } x \text{ ist Primzahl}\}$

Die Klasse NP

- NP steht für „nichtdeterministische Polynomialzeit“
- $L \in \text{NP} \iff_{\text{def}}$ es gibt ein Polynom p und ein $B \in \text{P}$, so dass für alle $x \in M$ gilt:

$$x \in L \iff \text{ es gibt } y \text{ mit } |y| \leq p(|x|) \text{ und } (x, y) \in B$$

- y heißt Lösung (Zeuge, Zertifikat) für x

Anschauung für Probleme in NP:

- Lösungen verifizieren ist „leicht“
- richtige Lösung finden ist „schwierig“

Die Klasse NP

Problem:	CLIQUE
Eingabe:	ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$
Frage:	Gibt es $U \subseteq V$ mit $\|U\| \geq k$ und $\{u, v\} \in E$ für alle $u, v \in U$ mit $u \neq v$?

CLIQUE liegt in NP:

- setze $p(n) =_{\text{def}} n$
- setze $B =_{\text{def}} \{(G, k, U) \mid U \text{ ist Clique in } G \text{ und } \|U\| \geq k\}$
- c_B kann in Zeit $O(n^2)$ berechnet werden $(n = |(G, k, U)|_1)$
- $(G, k) \in \text{CLIQUE} \iff$
  es gibt U mit $\|U\| = |U|_1 \leq |(G, k)|_1$ und $(G, k, U) \in B$

Die Klasse NP

Problem:	CLIQUE
Eingabe:	ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$
Frage:	Gibt es $U \subseteq V$ mit $\|U\| \geq k$ und $\{u, v\} \in E$ für alle $u, v \in U$ mit $u \neq v$?

Wie schnell lässt sich CLIQUE (deterministisch) lösen?

- Durchmusterung: auf Eingabe (G, k) durchlaufe alle Mengen U von k Knoten aus G und teste, ob U Clique in G ist
- Laufzeit hängt wesentlich von der Zahl der Knotenmengen ab: $\binom{n}{k}$
- $\binom{n}{k}$ maximal für $k = \lfloor n/2 \rfloor$: $\binom{n}{\lfloor n/2 \rfloor} = \Theta\left(\frac{2^n}{\sqrt{n}}\right) = \Theta(2^{n-\frac{1}{2}\log n})$

Das P-NP-Problem

Fakt: $\mathrm{P} \subseteq \mathrm{NP}$

- für $L \in \mathrm{P}$ definiere $B =_{\text{def}} L \times \{0,1\}$ und setze $p(n) = 1$
- $B \in \mathrm{P}$, denn: auf Eingabe (x, a) ignoriere a und wende Algorithmus für L auf x an
- Damit: $x \in L \iff (x, 0) \in B$
- Also liegt L auch NP

Frage: Gilt $\mathrm{P} = \mathrm{NP}$?

- bedeutendstes offenes Problem der Informatik und Mathematik
- steht auf Liste der Millenium-Probleme des Clay Mathematics Institute: 1.000.000 $ für Lösung
- Vermutung: $\mathrm{P} \neq \mathrm{NP}$

Reduktionen

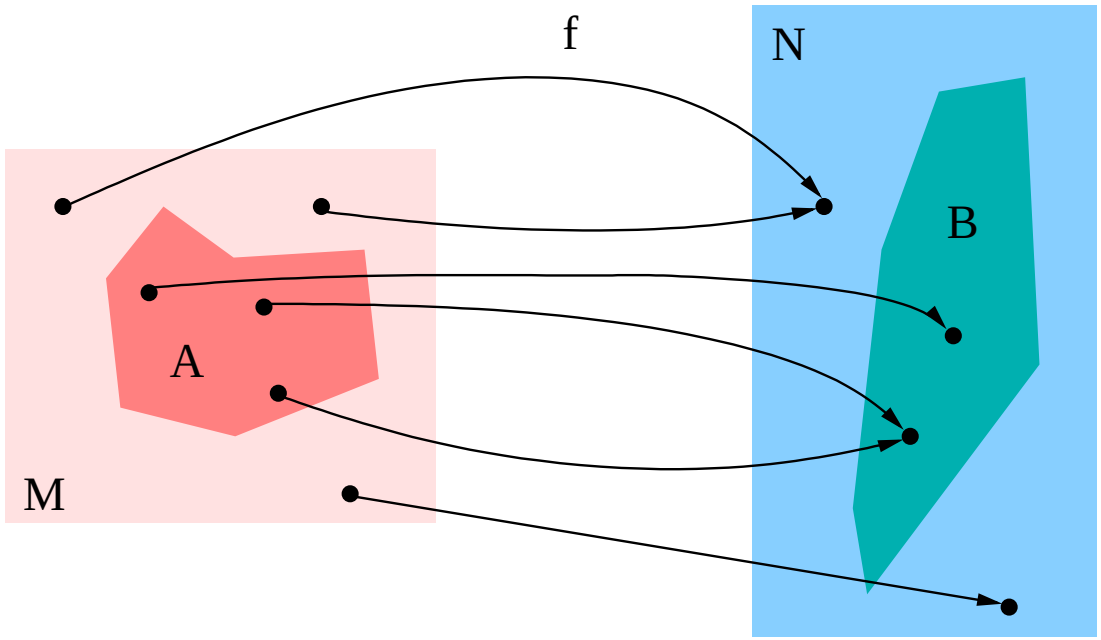
- Reduktionen definieren Relationen auf Entscheidungsproblemen
- in der Vorlesung: polynomielle Reduktionen
- $A \subseteq M$ in polynomieller Zeit auf $B \subseteq N$ reduzierbar $\iff_{\text{def}}$ es gibt eine Funktion $f : M \to N$ mit $f \in \text{FP}$, so dass für alle $x \in M$ gilt:

$$x \in A \iff f(x) \in B$$

- f heißt Reduktionsfunktion (i.A. weder injektiv noch surjektiv)
- Notation: $A \leq_m^p B$, falls A auf B in polynomieller Zeit reduzierbar
- Notation: $A \equiv_m^p B$, falls $A \leq_m^p B$ und $B \leq_m^p A$

Beachte: Reduktionsfunktion f i.A. weder injektiv noch surjektiv

Reduktionen



Reduktionsfunktion f für $A \leq_m^p B$

- f bildet M total auf N ab (unabhängig von A und B)
- jedes Element von A muss nach B abgebildet werden
- jedes Element von $\overline{A}$ muss nach $\overline{B}$ abgebildet werden

Reduktionen

Fakten:

- Relation $\leq_m^p$ ist reflexiv
- Relation $\leq_m^p$ ist transitiv
- Relation $\leq_m^p$ ist nicht antisymmetrisch
- Relation $\leq_m^p$ ist nicht symmetrisch
- Relation $\leq_m^p$ ist nicht total
- Relation $\equiv_m^p$ ist symmetrisch (damit: Äquivalenzrelation)

- Ist $A \leq_m^p B$ und ist $B \in \mathrm{P}$, so ist $A \in \mathrm{P}$
- Ist $A \leq_m^p B$ und ist $B \in \mathrm{NP}$, so ist $A \in \mathrm{NP}$

Reduktionen

Fakt: Hintereinanderausführung zweier Polynomialzeit-Algorithmen ist wieder ein Polynomialzeit-Algorithmus

- A mit Laufzeit $t_A^{\max}(n) \leq p_A(n)$
- B mit Laufzeit $t_B^{\max}(n) \leq p_B(n)$
- Algorithmus $A \circ B$: wende A auf Eingabe x an mit Ergebnis y und wende dann B auf Eingabe y an
- Laufzeit für $A \circ B$:

$$\begin{aligned} t_{A\circ B}^{\max}(n) &= O(t_A^{\max}(n) + t_B^{\max}(t_A^{\max}(n))) \\ &= O(\underbrace{p_A(n) + p_B(p_A(n))}_{p_{A\circ B}(n)}) \\ &= O(p_{A\circ B}(n)) \end{aligned}$$

NP-Vollständigkeit

- L ist NP-schwer $\iff_{\text{def}}$ für alle $A \in \text{NP}$ gilt $A \leq_m^p L$
- L ist NP-vollständig $\iff_{\text{def}}$ L ist NP-schwer und $L \in \text{NP}$
- NP-vollständige Probleme sind „schwierigste“ Probleme in NP

Fakt: Liegt ein NP-vollständiges Problem in P, so gilt $\text{P} = \text{NP}$

- m.a.W.: Ist $\text{P} \neq \text{NP}$, so liegt kein NP-vollständiges Problem in P
- Nachweis der NP-Vollständigkeit für konkretes Problem schließt polynomiellen Algorithmus für dieses Problem aus (es sei denn $\text{P} = \text{NP}$)
- Methodologie für Nachweis: Wähle irgendein NP-vollständiges Problem, welches dem gegebenen Problem ähnlich ist, und zeige Reduktion auf gegebenes Problem

Frage: Welches ist das erste NP-vollständige Problem?

Erfüllbarkeit aussagenlogischer Formeln

Menge L_{AL} der aussagenlogischen (booleschen) Formeln (über $\wedge, \vee, \neg$):

- für jede Variable x_i ist $x_i \in L_{\mathrm{AL}}$
- für $H_1, H_2 \in L_{\mathrm{AL}}$ sind $(H_1 \wedge H_2), (H_1 \vee H_2), \neg H_1 \in L_{\mathrm{AL}}$

- $\neg((x_1 \wedge x_2) \wedge (x_2 \wedge (\neg x_3 \vee (x_2 \wedge \neg x_1)))) \in L_{\mathrm{AL}}$
- $((x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)) \wedge ((\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))) \in L_{\mathrm{AL}}$
- $x_3 \wedge \wedge x_3 \notin L_{\mathrm{AL}}$

Erfüllbarkeit aussagenlogischer Formeln

- konjunktive Normalform (KNF, CNF) für boolesche Formel H:

$$H = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{m_i} L_{ij},$$

  wobei L_{ij} eine Variable x_ℓ oder eine negierte Variable $\neg x_\ell$ ist
- die L_{ij} heißen Literale
- die $\bigvee_{j=1}^{m_i} L_{ij}$ heißen Klauseln

- $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$ ist KNF mit 4 Klauseln zu je 2 Literalen

Erfüllbarkeit aussagenlogischer Formeln

Interessieren uns für die Wahrheitswerte aussagenlogischer Formeln:

x	y	AND(x, y)
0	0	0
0	1	0
1	0	0
1	1	1

x	y	OR(x, y)
0	0	0
0	1	1
1	0	1
1	1	1

x	NOT(x)
0	1
1	0

Für Formel $H \in L_{\mathrm{AL}}$ (mit den Variablen $x_1, x_2, \ldots, x_n$) definiere:

- Belegung von H ist Abbildung $I : \{x_1, x_2, \ldots, x_n\} \to \{0, 1\}$
- Ist $H = x_i$, so definiere $I(H) =_{\mathrm{def}} I(x_i)$
- Ist $H = (H_1 \wedge H_2)$, so definiere $I(H) =_{\mathrm{def}} \mathrm{AND}\big(I(H_1), I(H_2)\big)$
- Ist $H = (H_1 \vee H_2)$, so definiere $I(H) =_{\mathrm{def}} \mathrm{OR}\big(I(H_1), I(H_2)\big)$
- Ist $H = \neg H_1$, so definiere $I(H) =_{\mathrm{def}} \mathrm{NOT}\big(I(H_1)\big)$

Erfüllbarkeit aussagenlogischer Formeln

Wir betrachten die aussagenlogische Formel:

$$H =_{\text{def}} \neg(\underbrace{(x_1 \wedge x_2)}_{H_1} \wedge \underbrace{(x_2 \wedge (\neg x_3 \vee (x_2 \wedge \neg x_1)))}_{H_2})$$

Als Wertetabelle erhalten wir damit:

$I(x_1)$	$I(x_2)$	$I(x_3)$	$I(H_1)$	$I(H_2)$	$I(H)$
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	1	0	1

Beachte: Eine Belegung entspricht nur einer Zeile der Tabelle

Erfüllbarkeit aussagenlogischer Formeln

Erfüllbarkeit von Formeln H (mit Variablen $x_1, x_2, \ldots, x_n$):

- H heißt erfüllbar $\Longleftrightarrow$ es gibt eine Belegung $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ mit $I(H) = 1$

- $\neg((x_1 \wedge x_2) \wedge (x_2 \wedge (\neg x_3 \vee (x_2 \wedge \neg x_1))))$ ist erfüllbar
- $((x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)) \wedge ((\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)))$ ist nicht erfüllbar

Erfüllbarkeit aussagenlogischer Formeln

Problem:	SATISFIABILITY
Eingabe:	Menge $\{x_1, \ldots, x_n\}$ boolescher Variablen, boolesche Formel $H(x_1, \ldots, x_n)$ in KNF
Frage:	Ist H erfüllbar?

Theorem: [Cook 1971, Karp 1972, Levin 1973]
SATISFIABILITY ist NP-vollständig

Ausgewählte NP-vollständige Probleme

Problem:	CLIQUE
Eingabe:	ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$
Frage:	Enthält G Clique der Größe k?

Problem:	k-COLORABILITY
Eingabe:	ungerichteter Graph $G = (V, E)$
Frage:	Ist G mit k Farben (zulässig) färbbar?

Ausgewählte NP-vollständige Probleme

Problem:	Hamilton Circuit
Eingabe:	ungerichteter Graph $G = (V, E)$
Frage:	Gibt es einen Hamilton-Kreis in G?

Problem:	Travelling Salesperson
Eingabe:	ungerichteter Graph $G = (V, E)$ mit Kantengewichten $w : E \to \mathbb{R}_+$, Knotenmenge $U \subseteq V$, Zahl $k \in \mathbb{N}$
Frage:	Gibt es einen Kreis in G mit Gesamtgewicht höchstens k, der alle Knoten U besucht?

Ausgewählte NP-vollständige Probleme

Problem:	SUBSET SUM
Eingabe:	Zahlen $a_1, \ldots, a_n, b \in \mathbb{N}$
Frage:	Gibt es $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} a_i = b$?

Problem:	KNAPSACK
Eingabe:	n Gegenstände mit Gewichten $g_1, \ldots, g_n \in \mathbb{N}$ und Werten $v_1, \ldots, v_n \in \mathbb{N}$, Parameter $G, W \in \mathbb{N}$
Frage:	Gibt es $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} g_i \leq G$ und $\sum_{i \in I} v_i \geq W$?

Ausgewählte NP-vollständige Probleme

Problem:	Quadratic Diophantine Equations
Eingabe:	Zahlen $a, b, c \in \mathbb{N}$
Frage:	Gibt es $x, y \in \mathbb{N}$ mit $ax^2 + by = c$?

Problem:	Stable Triple Marriage
Eingabe:	gleich große Mengen A, B, C mit Personen dreier Geschlechter, Sympathiemenge $H \subseteq A \times B \times C$ (wobei $(a, b, c) \in H$ bedeutet, dass a, b und c sich eventuell heiraten würden)
Frage:	Gibt es ein Arrangement mit $\|A\|$ Heiraten (ohne Polygamie) ? (M.a.W.: Gibt es bijektive Funktionen $s : A \to B$ und $r : A \to C$ mit $(a, s(a), r(a)) \in H$ für alle $a \in A$?)

Beispielreduktion

Wollen zeigen: $\textsc{Satisfiability} \leq_m^p \textsc{Clique}$

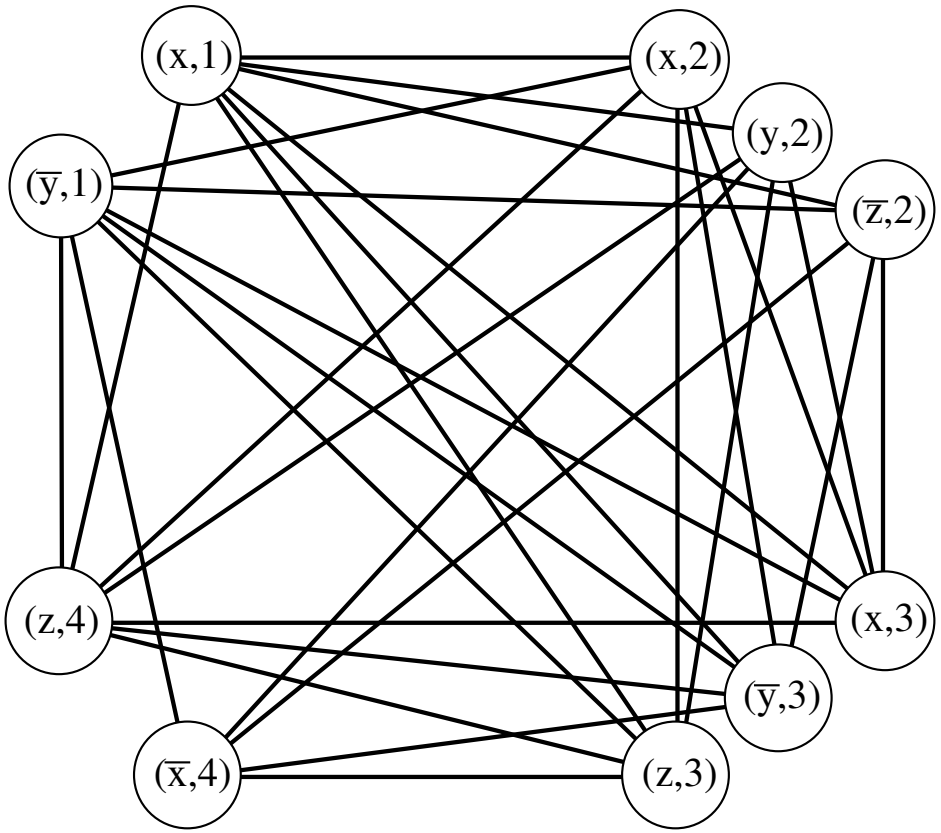
- es sei H eine Formel in KNF mit k Klauseln $C_1, \ldots, C_k$
- konstruieren Graph $G_H = (V_H, E_H)$ wie folgt:

$$V_H =_{\text{def}} \{ (L, i) \mid i \in \{1, \ldots, k\} \text{ und } L \text{ ist ein Literal in Klausel } C_i \}$$
$$E_H =_{\text{def}} \{ \{(L, i), (L', j)\} \mid i \neq j \text{ und } L \neq \neg L' \}$$

- Graph G_H kann in polynomieller Zeit aus Formel H berechnet werden
- definiere Reduktionsfunktion als

$$f(H) =_{\text{def}} (G_H, k)$$

Beispielreduktion



Graph G_H für $H =_{\text{def}} (x \vee \neg y) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee z)$
(beachte: $\overline{x}$ steht für $\neg x$)

Beispielreduktion

Fakt: H erfüllbar $\Longleftrightarrow$ G_H enthält Clique der Größe k

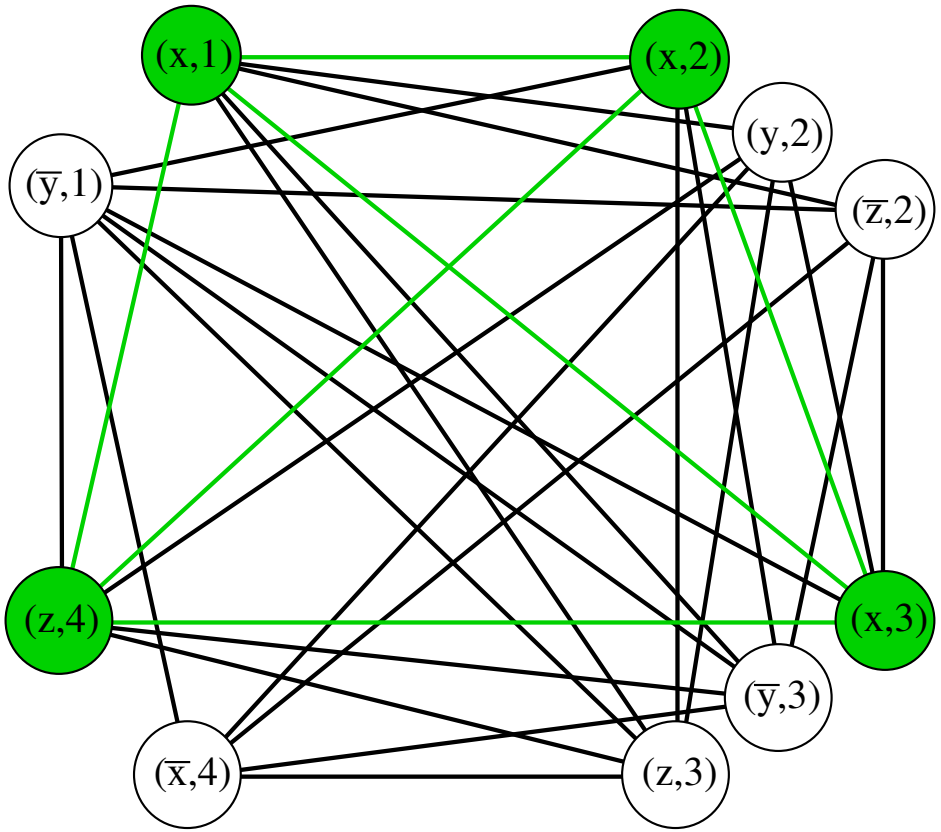
H ist erfüllbar (mittels Belegung I):

- dann gilt $I(C_1) = \cdots = I(C_k) = 1$ und in jeder Klausel wird zumindest ein Literal auf 1 gesetzt, seien $L_1, \ldots, L_k$ solche Literale
- dann gilt $L_i \neq \neg L_j$ für $i \neq j$ (sonst: $1 = I(x) = \text{NOT}\big(I(x)\big) = 0$)
- damit ist $\{(L_1, 1), \ldots, (L_k, k)\}$ Clique der Größe k in G_H

G_H enthält Clique der Größe k (nämlich $U \subseteq V_H$):

- für alle $i \in \{1, \ldots, k\}$ gibt es genau ein L_i mit $(L_i, i) \in U$
- da $L_i \neq \neg L_j$ für $i \neq j$, setze I so, dass $I(L) = 1$ für alle $(L, i) \in U$
- damit: $I(C_1) = \cdots = I(C_k) = 1$, also ist H erfüllbar

Beispielreduktion



Graph G_H für $H =_{\text{def}} (x \vee \neg y) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee z)$

- $I(x) = I(y) = I(z) = 1$ ist eine erfüllende Belegung für H
- $\{(x,1),(x,2),(x,3),(z,4)\}$ ist eine Clique der Größe 4 in G_H