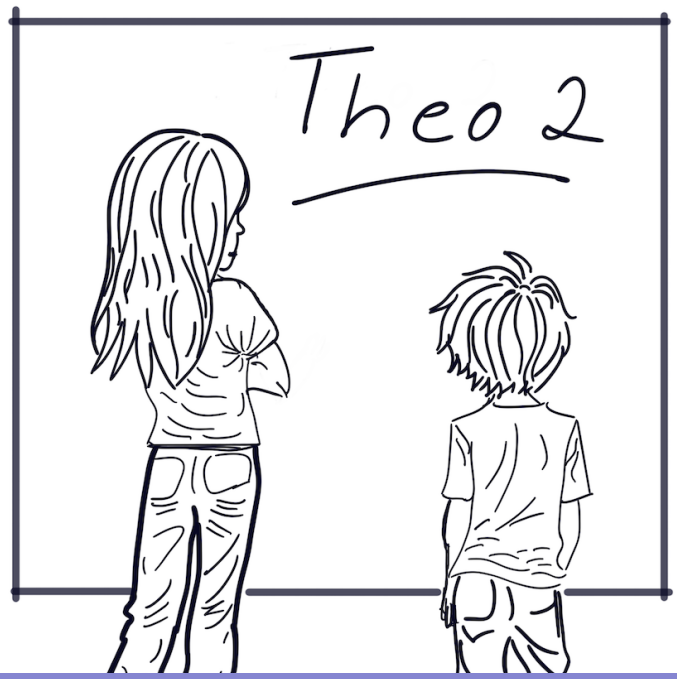
Theoretische Informatik 2

SoSe 2024

Berechenbarkeit und Komplexität

Prof. Dr. Sebastian Siebertz
AG Theoretische Informatik
MZH, Raum 3160
siebertz@uni-bremen.de

Universität Bremen



Komplexität

- Berechenbarkeit: was ist mit endlichen Ressourcen berechenbar?
  - Turingmaschinen
    - ★ Deterministisch oder nichtdeterministisch
  - Grammatiken
  - WHILE-Programme
- Komplexität: was ist effizient berechenbar?
  - Nicht beliebig viel Zeit und beliebig viel Speicher verfügbar!
  - Welches Rechenmodell wählen wir?

Komplexität

- Laufzeit eines Algorithmus: gemessen bzgl. Eingabelänge $T(n)$
  - Analyse $T(x)$ für jedes $x \in \Sigma^*$ ist zu detailliert.
- Was soll effizient bedeuten?
  - Abgrenzung polynomiell vs exponentiell:

n	n^3	2^n
5	125	32
10	1000	1024
20	8000	1048576
50	125000	1125899906842624

Komplexität

- Welches Rechenmodell?

Erweiterte Church-Turing These

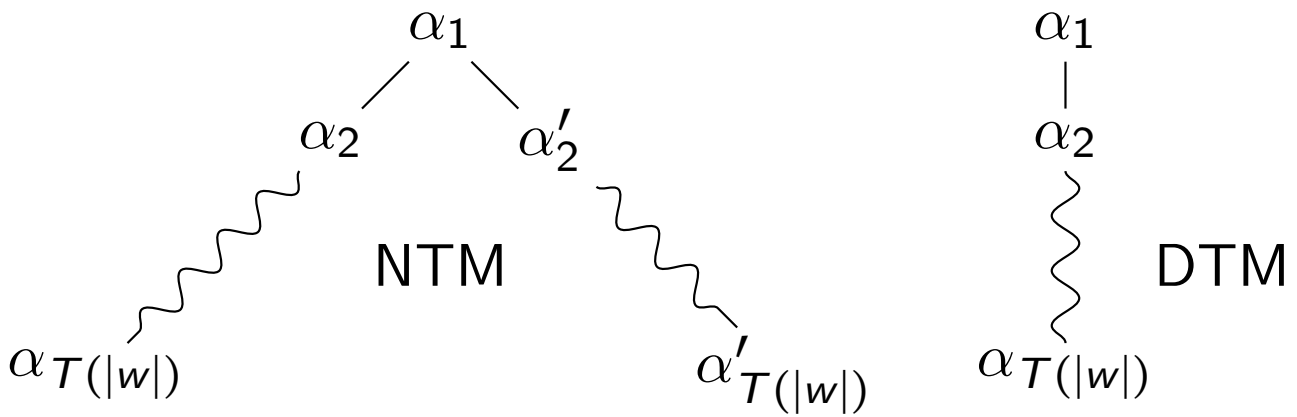
Für je zwei physikalisch realisierbare Rechnermodelle R_1 und R_2 gibt es ein Polynom $p(x, y)$, sodass t Rechenschritte im Modell R_1 bei Eingabe der Länge n durch $p(t, n)$ Rechenschritte im Modell R_2 simuliert werden können.

$\Rightarrow$ Bis auf polynomielle Faktoren spielt die Wahl des Modells keine Rolle.

- Wir arbeiten mit deterministischen Turingmaschinen.
- Nichtdeterministische Turingmaschinen werden trotzdem eine große Rolle spielen!

Komplexität

- Sei $T: \mathbb{N}_0 \to \mathbb{N}_0$ eine Funktion.
- Sei M eine (deterministische oder nichtdeterministische) Turingmaschine über Alphabet Σ.
- M heißt $T(n)$-zeitbeschränkt falls
  - für alle $w \in \Sigma^*$,
  - jede Berechnung der Maschine M auf Eingabe w nach höchstens $T(|w|)$ Schritten terminiert.



Zeitkomplexitätsklassen

- Sei $T: \mathbb{N}_0 \to \mathbb{N}_0$ eine Funktion. Wir definieren die Zeitkomplexitätsklassen
  - $\mathrm{DTime}(T(n)) :=$
    $\{L : \text{es ex. } T(n)\text{-zeitbeschränkte DTM } M \text{ mit } L(M) = L\}$,
  - $\mathrm{NTime}(T(n)) :=$
    $\{L : \text{es ex. } T(n)\text{-zeitbeschränkte NTM } M \text{ mit } L(M) = L\}$.
- Sei $T: \mathbb{N}_0 \to \mathbb{N}_0$ eine Funktion. Wir definieren
  - $\mathrm{FTime}(T(n)) :=$
    $\{f : \text{es ex. } T(n)\text{-zeitbeschränkte DTM } M, \text{ die } f \text{ berechnet}\}$.

P, NP und ExpTime

- Einige der wichtigsten Komplexitätsklassen:
  - Polynomialzeitberechenbare Probleme.

    $$\mathsf{P} := \bigcup_{p \text{ Polynom in } n} \mathsf{DTime}(p(n))$$

  - Nichtdeterministisch polynomialzeitberechenbare Probleme

    $$\mathsf{NP} := \bigcup_{p \text{ Polynom in } n} \mathsf{NTime}(p(n))$$

  - Exponentialzeitberechenbare Probleme

    $$\mathsf{ExpTime} := \bigcup_{p \text{ Polynom in } n} \mathsf{DTime}(2^{p(n)})$$

P, NP und ExpTime

- Probleme in P: theoretisch effizient lösbar.
  - Praktisch ist Laufzeit n^{1000} nicht effizient...
  - ...asymptotisch aber trotzdem verschwindend klein gegenüber 2^n.

Satz

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{ExpTime}.$$

Beweis.

- $\mathsf{P} \subseteq \mathsf{NP}$ ✓
- $\mathsf{NP} \subseteq \mathsf{ExpTime}$:
  - Durchsuche den Berechnungsbaum der NTM nach akzeptierender Konfiguration.
  - Dieser Baum hat nur exponentielle Größe.

P, NP, ExpTime

- Später:

$$P \neq \text{ExpTime}.$$

- Also $P \neq NP$ oder $NP \neq \text{ExpTime}$.
- Vermutung: Beide Ungleichungen gelten.
- Die Frage, ob

$$P = NP \text{ oder } P \neq NP?$$

gilt als eine der größten offenen Fragen der Informatik.

Vorsicht: NP steht nicht für nicht-polynomiell.

FP, FExpTime

- Polynomialzeitberechenbare Funktionen

  $$\mathrm{FP} := \bigcup_{p \text{ Polynom in } n} \mathrm{FTime}(p(n))$$

- Exponentialzeitberechenbare Funktionen

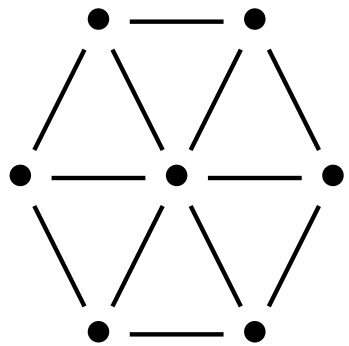
  $$\mathrm{FExpTime} := \bigcup_{p \text{ Polynom in } n} \mathrm{FTime}(2^{p(n)})$$

Beispiele

- Sort: Gegeben k Zahlen $n_1, \ldots, n_k \in \mathbb{N}$, sortiere die Zahlen.
- Formal: Funktion f mit Eingabe $\langle n_1, \ldots, n_k \rangle \in \{0,1\}^*$
  - $\langle n_1, \ldots, n_k \rangle$ Kodierung von $n_1, \ldots, n_k$, z.B. binäre Kodierung
- und Ausgabe $\left\langle n_{\pi(1)}, \ldots, n_{\pi(k)} \right\rangle$
  - π: Permutation, so dass $n_{\pi(1)} \leq \ldots \leq n_{\pi(k)}$.
- Sort $\in$ FP
  - z.B. Mergesort sortiert die Zahlen in Zeit $\mathcal{O}(k \log k)$.
  - Diese Laufzeit kann mit einer Turingmaschine nicht realisiert werden. Aber polynomielle Zeit reicht aus.

Beispiele

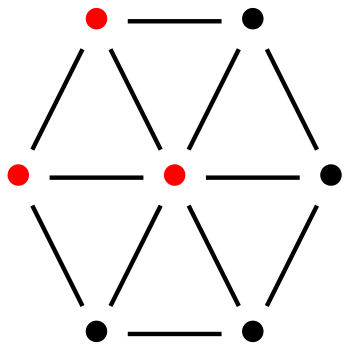
- Connectivity: Gegeben ein ungerichteter Graph G, ist G zusammenhängend?



- Formal: $\{\langle G \rangle \ : \ G \text{ Graph, } G \text{ zusammenhängend}\}$
  - $\langle G \rangle$ Kodierung von Graphen, z.B. die Adjazenzmatrix als String kodiert.
- Connectivity $\in$ P: z.B. mit einer Breitensuche in Zeit $\mathcal{O}(|V(G)| + |E(G)|)$.

Beispiele

- Clique: Gegeben ein Graph G und $k \in \mathbb{N}$. Enthält G eine Clique der Größe k?



- Clique $\in$ NP:
  - NTM schreibt zunächst nichtdeterministisch k Knoten auf ein zweites Band und
  - verifiziert dann, dass die geratenen Knoten paarweise adjazent sind.

Beispiele

- SAT: Gegeben eine Formel φ der Aussagenlogik, ist φ erfüllbar (engl. **sat**isfiable)?
- Menge AL der aussagenlogischen Formeln über der Variablenmenge $Var = \{X_1, X_2, \ldots\}$ ist induktiv definiert:
  - $0, 1 \in AL$ (die Booleschen Konstanten sind Formeln),
  - $Var \subseteq AL$ (jede Aussagenvariable ist eine Formel), und
  - Wenn $\varphi, \psi \in AL$, dann auch $\neg\psi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ und $(\varphi \to \psi) \in AL$.
- Beispiel:

$$(X_1 \vee \neg X_2) \wedge (X_1 \vee X_2) \wedge (\neg X_2 \vee \neg X_1)$$

Aussagenlogik

- Eine Belegung der Variablen ist eine Abbildung $\mathfrak{I}\colon \mathit{Var} \to \{0,1\}$.
- Belegung weist Formel φ einen Wahrheitswert $[\![\varphi]\!]^{\mathfrak{I}} \in \{0,1\}$ zu.
  - $[\![0]\!]^{\mathfrak{I}} = 0$ und $[\![1]\!]^{\mathfrak{I}} = 1$,
  - $[\![X]\!]^{\mathfrak{I}} = \mathfrak{I}(X)$ für $X \in \mathit{Var}$,
  - $[\![\neg\varphi]\!]^{\mathfrak{I}} = 1 - [\![\varphi]\!]^{\mathfrak{I}}$,
  - $[\![\varphi \wedge \psi]\!]^{\mathfrak{I}} = \min\{[\![\varphi]\!]^{\mathfrak{I}}, [\![\psi]\!]^{\mathfrak{I}}\}$,
  - $[\![\varphi \vee \psi]\!]^{\mathfrak{I}} = \max\{[\![\varphi]\!]^{\mathfrak{I}}, [\![\psi]\!]^{\mathfrak{I}}\}$ und
  - $[\![\varphi \to \psi]\!]^{\mathfrak{I}} = [\![\neg\varphi \vee \psi]\!]^{\mathfrak{I}} = \max\{1 - [\![\varphi]\!]^{\mathfrak{I}}, [\![\psi]\!]^{\mathfrak{I}}\}$.
- Eine Belegung ist eine erfüllende Belegung von φ, falls $[\![\varphi]\!]^{\mathfrak{I}} = 1$.
  - Wir schreiben auch $\mathfrak{I} \vDash \varphi$.
- Eine Formel φ ist erfüllbar, wenn es eine erfüllende Belegung gibt.

Beispiele

$$(X_1 \vee \neg X_2) \wedge (X_1 \vee X_2) \wedge (\neg X_2 \vee \neg X_1)$$

ist erfüllbar, z.B. die Belegung $\mathfrak{I}$ mit $\mathfrak{I}(X_1) = 1$ und $\mathfrak{I}(X_2) = 0$ ist erfüllende Belegung.

- SAT $\in$ NP:
  - NTM schreibt nichtdeterministisch erfüllende Belegung auf zweites Band und
  - verifiziert dann, dass diese tatsächlich erfüllend ist.

NP = polynomialzeit-verifizierbare Lösungen

Satz

Ein Problem $L \subseteq \Sigma^*$ liegt in NP genau dann, wenn ein Polynom p und eine polynomialzeitbeschränkte deterministische Turingmaschine M existieren (genannt Verifizierer), so dass für jedes $v \in \Sigma^*$ gilt

$$v \in L \Leftrightarrow \text{es ex. } w \in \Sigma^{p(|v|)}, \text{ so dass } M \text{ das Wort } v\#w \text{ akzeptiert.}$$

Das Wort w der Länge $p(|v|)$ wird ein Zertifikat für v genannt.

Ein Problem L liegt in NP genau dann,

wenn Lösungen für L in Polynomialzeit verifiziert werden können.

NP = polynomialzeit-verifizierbare Lösungen

- Beispiele:
  - Für Clique ist das Zertifikat die Kodierung einer Clique der Größe k.
  - Für SAT ist das Zertifikat die Kodierung einer erfüllenden Belegung.

Beweis

$\Rightarrow$

- Es sei $L \in$ NP und sei N $p(n)$-zeitbeschränkte NTM für Polynom p, die L entscheidet.
- Für $v \in L$ existiert akzeptierende Berechnung $\alpha_1 \vdash_N \ldots \vdash_N \alpha_m$ von N auf v mit Länge $m \leq p(|v|)$.
- Zertifikat für v: das Wort $\alpha_1 \vdash \ldots \vdash \alpha_m$.
- DTM M verifiziert bei Eingabe $v\#\alpha_1 \vdash \ldots \vdash \alpha_m$, dass dies eine akzepierende Berechnung von N auf v ist (in Polynomialzeit).

Beweis

$\Leftarrow$

- Sei L Sprache, so dass ein Polynom $p\colon \mathbb{N} \to \mathbb{N}$ und polynomialzeitbeschränkte deterministische Turingmaschine M existieren, so dass für jedes $v \in \Sigma^*$ gilt

  $v \in L \Leftrightarrow$ es ex. $w \in \Sigma^{p(|v|)}$, so dass M das Wort $v\#w$ akzeptiert.

- NTM für L schreibt bei Eingabe $v \in \Sigma^*$ nichtdeterministisch ein solches Wort $w \in \Sigma^{p(|v|)}$ hinter die Eingabe v und simuliert dann M auf vw.

Polynomialzeitreduktionen und NP-schwere Probleme

- Wir vermuten, dass $P \neq NP$, haben aber noch keinen Beweis gefunden.
- Es gibt hunderte von wichtigen Problemen in NP, für die wir keinen Polyomialzeitalgorithmus kennen.
- Wie zeigen wir, dass ein Problem schwer ist?
  - Es gibt schließlich auch leichte Probleme in NP, da $P \subseteq NP$...
- Führen Polynomialzeitreduktionen ein, analog zu Reduktionen in der Berechenbarkeitstheorie.
  - Schreibe $L_1 \leq_p L_2$, wenn L_1 in Polynomialzeit auf L_2 reduzierbar ist.
  - Wenn L_2 in Polynomialzeit gelöst werden kann, dann kann auch L_1 in Polynomialzeit gelöst werden.
  - Die schwersten Probleme in NP: $L' \leq_p L$ für alle Probleme $L' \in NP$:

NP-schwere Probleme

Polynomialzeitreduktionen

- Eine Reduktion f von $L_1 \subseteq \Sigma^*$ auf $L_2 \subseteq \Sigma^*$ heißt Polynomialzeitreduktion, wenn es ein Polynom p und eine $p(n)$-zeitbeschränkte DTM gibt, die f berechnet.
- Wenn eine Polynomialzeitreduktion von L_1 auf L_2 existiert, dann schreiben wir $L_1 \leq_p L_2$.

Polynomialzeitreduktionen

Lemma

Es gelte $L_1 \leq_p L_2$. Wenn $L_2 \in \mathrm{P}$, dann auch $L_1 \in \mathrm{P}$.

Beweis.

- $L_1 \leq_p L_2$: es existiert $p(n)$-zeitberechenbare Funktion $f\colon \Sigma^* \to \Sigma^*$, so dass für alle $w \in \Sigma^*$ gilt

$$w \in L_1 \Leftrightarrow f(w) \in L_2.$$

- $L_2 \in \mathrm{P}$: es existiert DTM M_2, die L_2 in Zeit $q(n)$ entscheidet für Polynom q.

Beweis fortgesetzt

- Konstruiere DTM M_1, die
  - auf Eingabe $w \in \Sigma^*$ den Wert $f(w)$ berechnet und dann
  - M_2 auf $f(w)$ simuliert.
  - M_1 akzeptiert genau dann, wenn M_2 akzeptiert.
- M_1 akzeptiert w genau dann, wenn $f(w) \in L_2 \Leftrightarrow w \in L_1$ ✓
- Laufzeit von M_1: höchstens $p(|w|) + q(p(|w|))$.
  - Verkettung von Polynomen: wieder ein Polynom.

Polynomialzeitreduktionen und NP-Vollständigkeit

Lemma

Es gelte $L_1 \leq_p L_2 \leq_p L_3$. Dann gilt $L_1 \leq_p L_3$.

- Eine Sprache L heißt NP-schwer, wenn für alle $L' \in \text{NP}$ gilt: $L' \leq_p L$.
- L heißt NP-vollständig, wenn $L \in \text{NP}$ und L NP-schwer ist.

Satz

Für jede NP-schwere Sprache L gilt: wenn $L \in \text{P}$, dann $\text{P} = \text{NP}$.

Beweis.

- Angenommen $L \in \text{P}$ ist NP-schwer. Sei $L' \in \text{NP}$.
- Es gilt $L' \leq_p L$, also $L' \in \text{P}$ mit dem ersten Lemma.

Satz von Cook und Levin

Satz von Cook und Levin

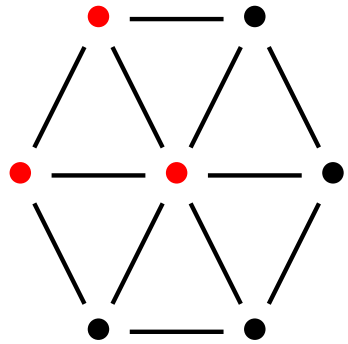
SAT ist NP-vollständig.

- 1971 von Stephen A. Cook bewiesen.
- 1973 unabhängig von Leonid Levin in der Sowjetunion bewiesen, im Westen über 10 Jahre lang nicht beachtet.
- 1978 emigrierte Levin in die USA.

Zum Beweis:

- Wir wissen bereits: SAT $\in$ NP.
- Wir müssen zeigen: für alle $L' \in$ NP gilt $L' \leq_p$ SAT.

Beispiel: Clique $\leq_p$ SAT



- Beispiel: Clique $\leq_p$ SAT.
- Zu Eingabe G, k konstruiere in Polynomialzeit aussagenlogische Formel $\varphi_{G,k}$, so dass

$$G \text{ enthält Clique der Größe } k \Leftrightarrow \varphi_{G,k} \text{ erfüllbar.}$$

- Polynomialzeit immer bezüglich Eingabegröße:
  - Größe der Kodierung $\langle G, k \rangle$.
  - Z.B. mit Adjazenzmatrix
  - Wenn $|V(G)| = n \quad \rightarrow \quad$ Eingabegröße $\mathcal{O}(n^2 + \lceil \log k \rceil)$.

Notation SAT

- Konjunktion und Disjunktion von Formeln ist assoziativ , d.h.

$$((\varphi \wedge \psi) \wedge \chi) \equiv (\varphi \wedge (\psi \wedge \chi))$$

und

$$((\varphi \vee \psi) \vee \chi) \equiv (\varphi \vee (\psi \vee \chi)).$$

- Notation $\varphi \equiv \psi$ bedeutet: φ und ψ sind äquivalent, d.h., für jede Belegung $\mathfrak{I}$ gilt $\mathfrak{I} \models \varphi \Leftrightarrow \mathfrak{I} \models \psi$.
- Schreibe einfach

$$\varphi \wedge \psi \wedge \chi.$$

- Allgemeiner für endliche Indexmenge I:
  - $\bigwedge_{i \in I} \varphi_i$ für Konjunktion und
  - $\bigvee_{i \in I} \varphi_i$ für die Disjunktion der Formeln.

Beispiel: Clique $\leq_p$ SAT

- Sei (G, k) Clique Instanz.
- Konstruiere die Formel $\varphi_{G,k}$:
  - Für jeden Knoten $v \in V(G)$: Variablen $X_{v,1}, \ldots, X_{v,k}$.
  - $\mathfrak{I}(X_{v,i}) = 1$ soll bedeuten, dass v der i-te Knoten der Clique ist.
  - Es darf höchstens ein Knoten als i-ter Knoten ausgewählt werden: für jeden Knoten v und jedes i die Teilformel

$$\psi_{v,i} = X_{v,i} \rightarrow \bigwedge_{u \neq v} \neg X_{u,i}.$$

  - Es soll mindestens ein Knoten ein i-ter Knoten ausgewählt werden: für jedes i die Teilformel

$$\chi_i = \bigvee_{v \in V(G)} X_{v,i}.$$

Beispiel: Clique $\leq_p$ SAT

- Sei

$$\sigma = \bigwedge_{v \in V(G)} \bigwedge_{1 \leq i \leq k} \psi_{v,i} \wedge \bigwedge_{1 \leq i \leq k} \chi_i.$$

Lemma

Die Formel σ ist erfüllbar und $\mathfrak{I}$ ist erfüllende Belegung genau dann, wenn für jedes $1 \leq i \leq k$ genau ein v existiert mit $\mathfrak{I}(X_{v,i}) = 1$.

- Wenn $\mathfrak{I}(X_{u,i}) = 1$ für ein $1 \leq i \leq k$, so sagen wir, dass Knoten u ausgewählt wird.
- Wenn Knoten u ausgewählt wird und $\{u, v\} \notin E(G)$, so darf Knoten v nicht ausgewählt werden:

$$\omega = \bigwedge_{\substack{\{u,v\} \notin E(G) \\ u \neq v}} \bigwedge_{i,j} \left(X_{u,i} \rightarrow \neg X_{v,j} \right).$$

Beispiel: Clique $\leq_p$ SAT

- Sei $\varphi_{G,k} = \sigma \wedge \omega$.
- Wir zeigen

$$G \text{ enthält Clique der Größe } k \Leftrightarrow \varphi_{G,k} \text{ erfüllbar.}$$

- Sei $C \subseteq V(G)$ eine Clique der Größe k in G.
  - Nummeriere die Knoten von C als $v_1, \ldots, v_k$.
  - Dann ist $\mathfrak{I}$ mit $\mathfrak{I}(X_{v_i,i})$ für $1 \leq i \leq k$ eine erfüllende Belegung von $\varphi_{G,k}$.
- Sei umgekehrt $\mathfrak{I}$ eine erfüllende Belegung von $\varphi_{G,k}$.
  - Nach Lemma werden von $\mathfrak{I}$ genau k Knoten ausgewählt. Sei C die Menge der ausgewählten Knoten.
  - Da $\mathfrak{I} \vDash \omega$ gilt für alle ausgewählten Knoten $u \neq v$ auch $\{u, v\} \in E(G)$.
  - Also ist C eine Clique.

Beispiel: Clique $\leq_p$ SAT

- $\psi_{v,i} = X_{v,i} \rightarrow \bigwedge_{u \neq v} \neg X_{u,i}$ hat Größe $\mathcal{O}(n \cdot k)$.
- $\chi_i = \bigvee_{v \in V(G)} X_{v,i}$ hat Größe $\mathcal{O}(n)$.
- $\sigma = \bigwedge_{v \in V(G)} \bigwedge_{1 \leq i \leq k} \psi_{v,i} \wedge \bigwedge_{1 \leq i \leq k} \chi_i$ hat Größe $\mathcal{O}(n^2 \cdot k^2 + n \cdot k) \subseteq \mathcal{O}(n^2 \cdot k^2)$.
- $\omega = \bigwedge_{\substack{\{u,v\} \notin E(G) \\ u \neq v}} \bigwedge_{i,j} \big(X_{u,i} \rightarrow \neg X_{v,j}\big)$ hat Größe $\mathcal{O}(n^2 \cdot k^2)$.
- Insgesamt hat die Formel also Größe $\mathcal{O}(n^2 \cdot k^2)$ und kann auch in Polynomialzeit berechnet werden.