Informatik III

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Christian Schindelhauer
Wintersemester 2006/07
20. Vorlesung
12.01.2007

Komplexitätstheorie - Zeitklassen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Die Komplexitätsklassen TIME**
  - DTIME, NTIME
  - P
  - NP
- **Das Cook-Levin-Theorem**
  - Polynomial-Zeit-Reduktion
  - Reduktion von 3SAT auf Clique
  - NP-vollständigkeit
  - SAT ist NP-vollständig
- **Weitere NP-vollständige Probleme**
  - Knotenüberdeckung (Vertex-Cover)
  - Das Hamiltonsche Pfadproblem
  - Das ungerichtete Hamiltonsche Pfadproblem
  - Das Teilsummenproblem

Der Verifizierer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

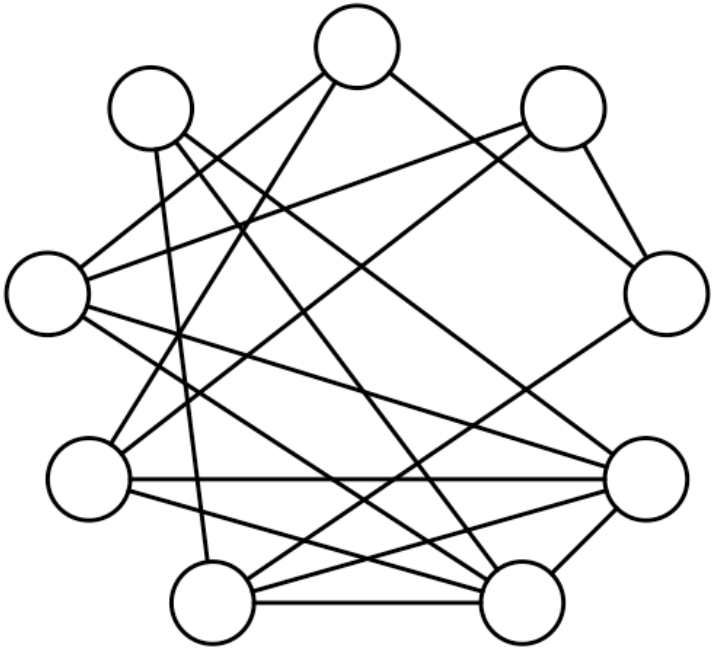
- **Definition**
  - Ein **Verifizierer** für eine Sprache A ist eine DTM V, wobei
    - A = {w | V akzeptiert <w,c> für ein Wort c}
  - Ein **Polynom-Zeit-Verifizierer** hat eine Laufzeit die durch ein Polynom $|w|^k$ beschränkt ist.
  - Eine Sprache ist in **Polynom-Zeit verifizierbar**, falls sie einen Polynom-Zeit-Verifizierer hat.

- **Theorem**
  - NP beschreibt genau die Sprachen, die in Polynom-Zeit verifiziert werden können.

Das Cliquen-Problem ist in NP

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

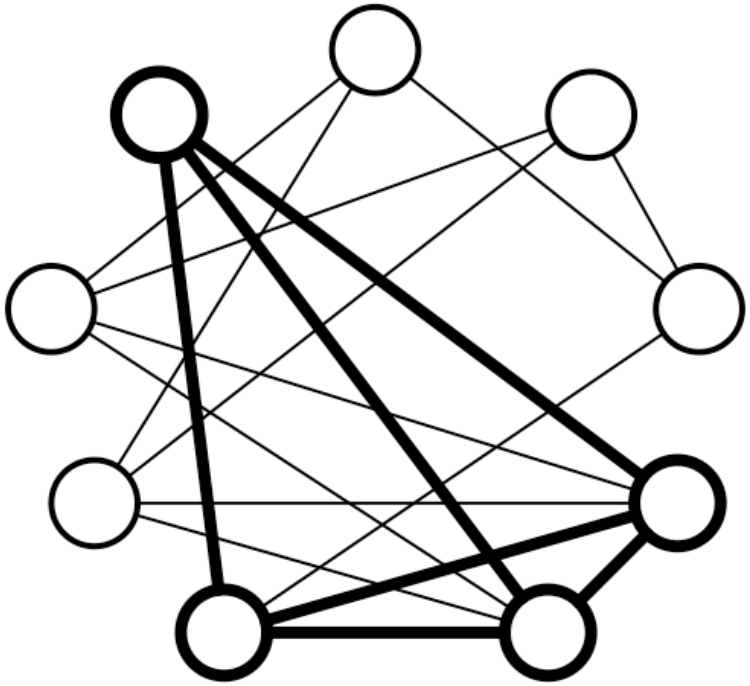
- **Definition k-Clique**
  - Ein Graph G=(V,E) hat eine k-Clique,
    - falls es k verschiedene Knoten gibt,
    - so dass jeder mit jedem anderen eine Kante in G verbindet
- **Das Cliquen-Problem**
  - Gegeben:
    - Ein ungerichteter Graph G
    - Eine Zahl k
  - Gesucht:
    - Hat der Graph G eine Clique der Größe k?



k=4

Das Cliquen-Problem ist in NP

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

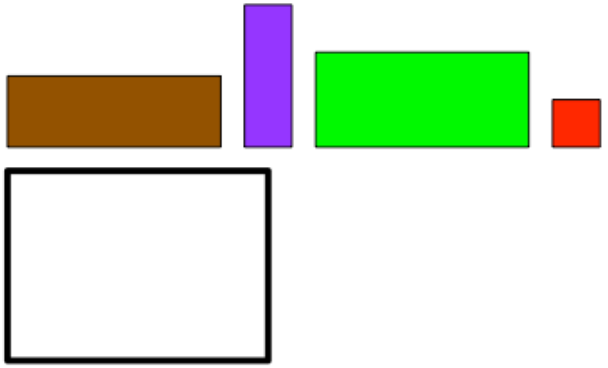


- **Definition k-Clique**
  - Ein Graph G=(V,E) hat eine k-Clique,
    - falls es k verschiedene Knoten gibt,
    - so dass jeder mit jedem anderen eine Kante in G verbindet
- **Das Cliquen-Problem**
  - Gegeben:
    - Ein ungerichteter Graph G
    - Eine Zahl k
  - Gesucht:
    - Hat der Graph G eine Clique der Größe k?
- **Theorem:**
  - Das Cliquen-Problem ist in NP
- **Beweis:**
  - Betrachte A = {<G,k> | es gibt Knoten $v_1,v_2,..,v_k$ die eine k-Clique in G sind}.
  - Verifizierer testet, ob die k Knoten unterschiedlich sind
  - und ob zwischen allen Knoten eine Kante existiert.
  - Laufzeit: $O(n^2)$

Das Koffer-Pack-Problem ist in NP

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition Koffer-Pack-Problem:**
  - Gegeben:
    - Einge Menge M von kleinen Rechtecken mit geradzahliger Kantenlänge
    - Ein großes Rechteck R
  - Gesucht:
    - Passen die kleinen Rechtecke orthogonal überschneidungsfrei in das große Rechteck?

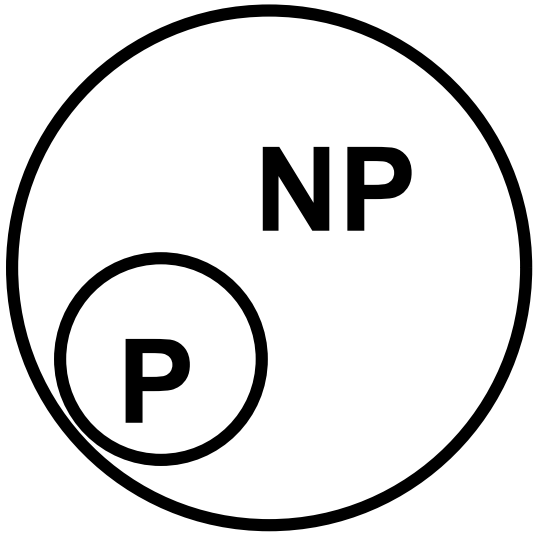


- **Theorem**
  - Das Koffer-Pack-Problem ist in NP
- **Beweis**
  - A = {<M,R>| es gibt eine Lagebeschreibung in der die Rechtecke sich nicht überschneiden und innerhalb von R liegen}

Die Frage P versus NP (I)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **P = Klasse aller Probleme, die effizient *entschieden* werden können**
- **NP = Klasse aller Problem, die effizient *verifiziert* werden können**
  - Beispiele: Hamiltonscher Pfad, Clique, Teilsummenproblem, Koffer-Pack-Problem
- **Es gibt jetzt zwei Möglichkeiten**

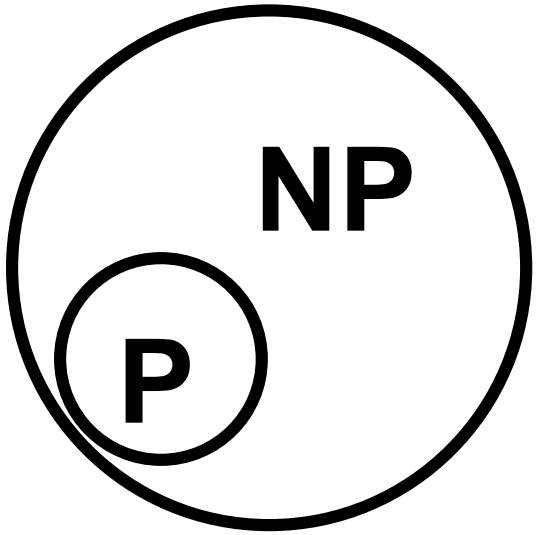


- **Was weiß man?**
  - **P ⊆ NP ⊆** $\bigcup_k \mathrm{TIME}(2^{n^k})$ **= EXPTIME**

Die Frage P versus NP (I)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **P = Klasse aller Probleme, die effizient *entschieden* werden können**
- **NP = Klasse aller Problem, die effizient *verifiziert* werden können**
  - Beispiele: Hamiltonscher Pfad, Clique, Teilsummenproblem, Koffer-Pack-Problem
- **Es gibt jetzt zwei Möglichkeiten**



- **Was weiß man?**
  - **P ⊆ NP ⊆** $\bigcup_k \mathrm{TIME}(2^{n^k})$ **= EXPTIME**

Die Frage P versus NP (II)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

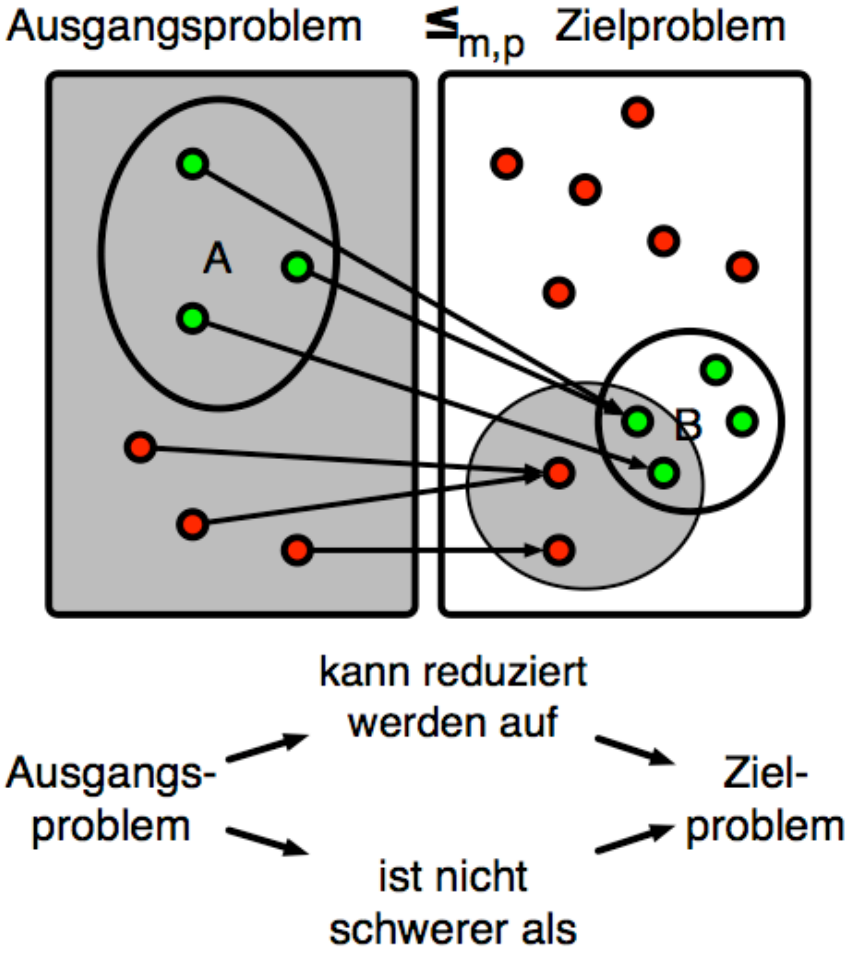
- **Was folgt aus P=NP?**
  - Schwierige kombinatorische Probleme sind lösbar
  - Viele kryptographische Systeme werden in Polynomzeit lösbar:
    - A = { Code | es gibt einen Originaltext der zum Code passt}
    - A' = { Code | es gibt einen Originaltext, der mit b_1 anfängt und zum Code passt}
    - A'' = { Code | es gibt einen Originaltext, der mit b_1 b_2 anfängt und zum Code passt}
    - ...
    - Wäre jeweils in P. Damit kann der Originaltext Bit für Bit aufgedröselt werden
- **Was folgt aus P$\neq$NP.**
  - Dann bleibt gewisse Probleme in NP praktisch unlösbar.

- **Was weiß man?**
  - Wenig, außer:
  - Cook-Levin: Genau dann wenn man ein bestimmtes Problem in Polynomzeit lösen kann, dann ist P=NP.

Die Polynom-Zeit-Abbildungsreduktion

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

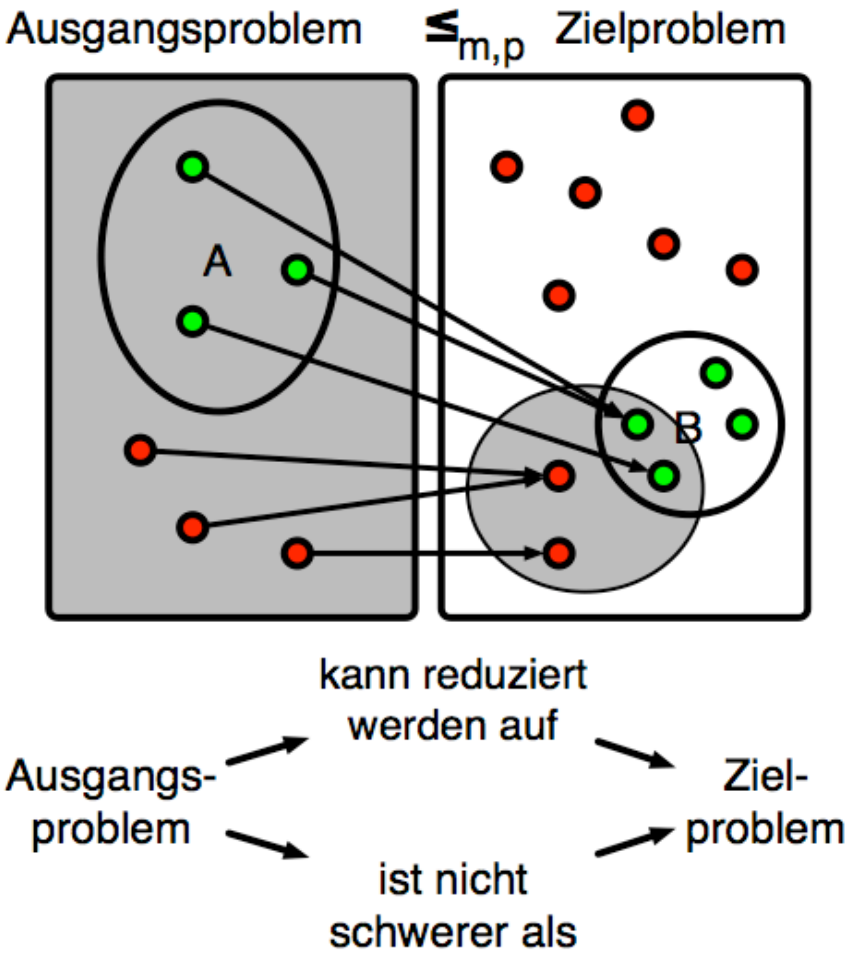
- **Definition (Abbildungsreduktion, Polynomial Time Mapping Reduction, Many-one)**
  - Eine Sprache A kann durch Abbildung auf eine Sprache B in Polynom-Zeit reduziert werden: $\mathbf{A \leq_{m,p} B}$,
    - falls es eine in Polynom-Zeit berechenbare Funktion $f: \Sigma^* \to \Sigma^*$ gibt,
    - so dass für alle w: $w \in A \Leftrightarrow f(w) \in B$
  - Die Funktion f heißt die **Reduktion** von A auf B.



Polynom-Zeit-Abbildungsreduktion und P

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

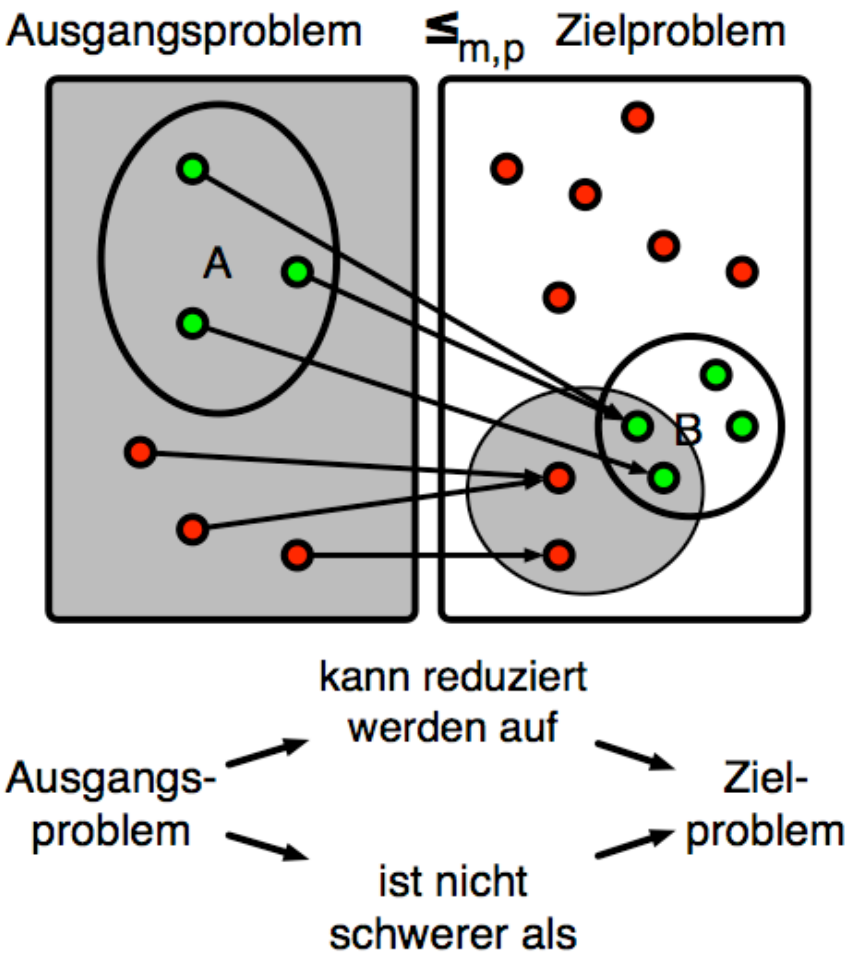
- **Theorem**
  - Falls $A \leq_m B$ und B ist in P, dann ist A auch in P.
- **Beweis**
  - Sei M, eine Turing-Maschine, die B in Polynom-Zeit $O(n^c)$ entscheidet.
  - Betrachte die Polynom-Zeit-TM:
  - N = "Auf Eingabe w:
    - Berechne f(w) in Zeit $O(n^k)$
    - Führe die Berechnung von M auf Eingabe f(w) durch
    - N gibt aus, was M ausgibt"
  - N entscheidet A
  - N berechnet A in Polynom-Zeit:
    - f(w) kann in Polynom-Zeit berechnet werden
    - f(w) hat Länge $O(|w|^k)$
    - M rechnet auf Eingabe f(w) in Zeit $O(|f(w)|^c) = O(|w|^{kc})$



Polynom-Zeit-Abbildungsreduktion und NP

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - Falls $A \leq_m B$ und B ist in NP, dann ist A auch in NP.
- **Beweis**
  - Sei M, eine NTM, die B in Polynom-Zeit $O(n^c)$ entscheidet.
  - Betrachte die Polynom-Zeit-NTM:
  - N = "Auf Eingabe w:
    - Berechne f(w) in Zeit $O(n^k)$
    - Führe die (nicht-deterministische) Berechnung von M auf Eingabe f(w) durch
    - N gibt aus, was M ausgibt"
  - N entscheidet A
    - da $w \in A \Leftrightarrow f(w) \in B$
  - N berechnet A in Polynom-Zeit:
    - s.o.



Wiederholung Boolesche Algebra

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Eine Boolesche Variable ist entweder falsch (0) oder wahr (1)**
- **Als Operationen für Boolesche Variablen verwenden wir**
  - die Disjunktion (Oder): $x \vee y$
    - $0 \vee 0 = 0$
    - $0 \vee 1 = 1$
    - $1 \vee 0 = 1$
    - $1 \vee 1 = 1$
  - die Konjunktion (Und): $x \wedge y$
    - $0 \wedge 0 = 0$
    - $0 \wedge 1 = 0$
    - $1 \wedge 0 = 0$
    - $1 \wedge 1 = 1$
  - die Negation $\neg x = \overline{x}$
    - $\neg 0 = 1$
    - $\neg 1 = 0$
- **Eine Boolesche Formel besteht aus der Kombination dieser Operationen**
  - $f(x,y) = (x \vee y) \wedge (\neg x \vee \neg y)$
- **Rechenregeln:**
  - Doppelte Negation
    - $\neg \neg x = x$
  - Kommuntativ-Gesetz
    - $x \wedge y = y \wedge x$
    - $x \vee y = y \vee x$
  - Distributiv-Gesetz
    - $(x \vee y) \wedge z = (x \wedge z) \vee (x \wedge z)$
    - $(x \wedge y) \vee z = (x \vee z) \wedge (x \vee z)$
  - De-Morgan-Regeln
    - $\neg (x \vee y) = \neg x \wedge \neg y$
    - $\neg (x \wedge y) = \neg x \vee \neg y$

Besondere Boolesche Funktionen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Literal:**
  - ist eine einfache oder negierte Variable
  - z.B.: x, y, z, ¬x, ¬y, ¬z
- **Klausel:**
  - ist eine Disjunktion von Literalen
  - z.B.: $x \vee y$, $z \vee \neg x \vee \neg y$, $x \vee \neg z$
- **Konjunktive Normalform (CNF)**
  - Konjunktion von Klauseln
  - z.B.: $(x \vee y) \wedge (z \vee \neg x \vee \neg y) \wedge (x \vee \neg z)$
- **k-konjunktive Normalform (k-CNF)**
  - Konjunktion von Klauseln aus k Literalen, z.B. 3-CNF
    - $(x \vee y \vee y) \wedge (z \vee \neg x \vee \neg y) \wedge (x \vee \neg z \vee \neg z)$
- **Disjunktive Normalform (DNF)**
  - Disjunktion aus Konjunktion von Literalen
    - $(x \wedge y) \vee (z \wedge \neg x \wedge \neg y) \vee (x \wedge \neg z)$

Das Erfüllbarkeitsproblem Boolescher Funktionen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Eine Boolesche Funktion $f(x_1,x_2,..,x_n)$ ist erfüllbar, wenn es eine Wertebelegung für $x_1,x_2,..,x_n$ gibt, so dass $f(x_1,x_2,..,x_n) = 1$**
  - $(x \vee y) \wedge (z \vee \neg x \vee \neg y) \wedge (x \vee \neg z)$ ist erfüllbar, da
    - die Belegung $x = 1, y = 0, z = 0$
  - $(1 \vee 0) \wedge (0 \vee 0 \vee 1) \wedge (1 \vee 1) = 1 \wedge 1 \wedge 1 = 1$ liefert.
- **Definition (Satisfiability Problem of Boolean Formulas)**
  - Das Erfüllbarkeitsproblem der Booleschen Funktion ist definiert als:
  - **SAT = { ϕ | ϕ ist eine erfüllbare Funktion}**, d.h.
  - **Gegeben**:
    - Boolesche Funktion ϕ
  - **Gesucht**:
    - Gibt es $x_1,x_2,..,x_n$ so dass $\phi(x_1,x_2,..,x_n) = 1$
- **Spezialfall: 3-SAT**
  - **3-SAT = { ϕ | ϕ ist eine erfüllbare Funktion in 3-CNF}**
  - **z.B.:** $\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$

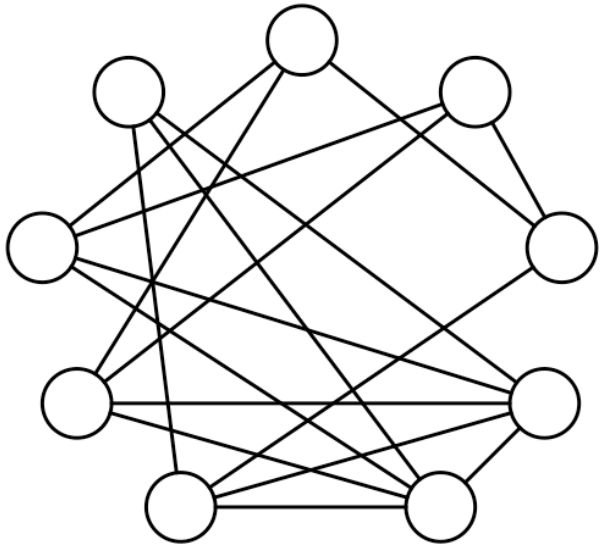
Das 3-SAT-Problem und das Clique-Problem

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **3-SAT:**
  - **Gegeben**:
    - Eine Boolesche Formel in 3-CNF
  - **Gesucht**:
    - Gibt es eine erfüllende Belegung
- **Definition k-Clique**
  - Ein ungerichteter Graph G=(V,E) hat eine k-Clique,
    - falls es k verschiedene Knoten gibt,
    - so dass jeder mit jedem anderen eine Kante in G verbindet
- **CLIQUE:**
  - Gegeben:
    - Ein ungerichteter Graph G
    - Eine Zahl k
  - Gesucht:
    - Hat der Graph G eine Clique der Größe k?

$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

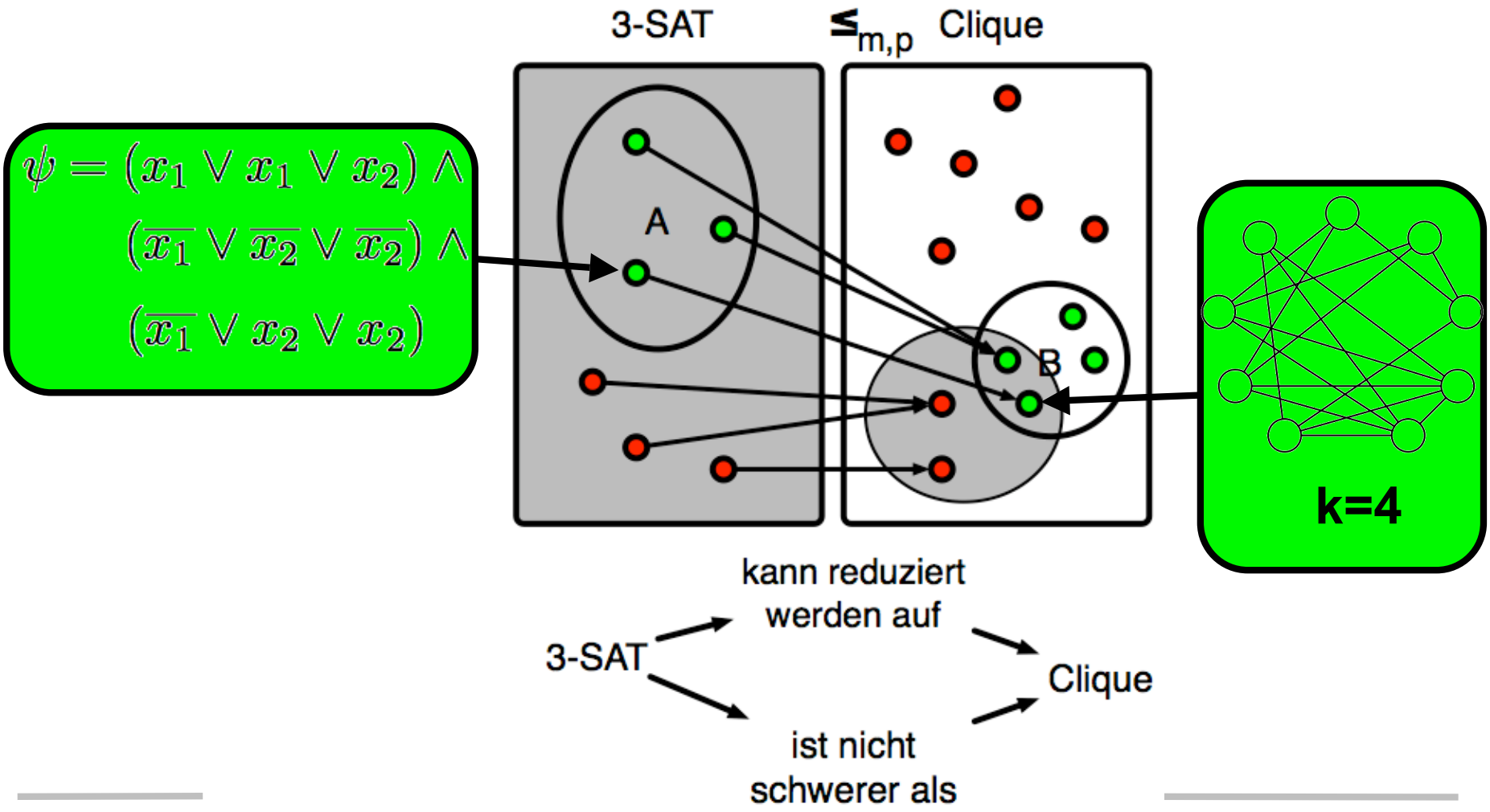
k=4



3-SAT läßt sich auf Clique reduzieren

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

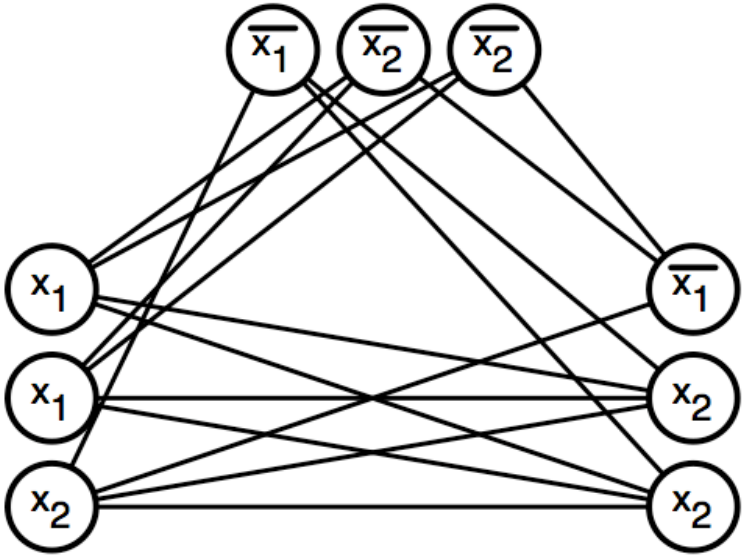
➢**Theorem: 3-SAT $\leq_{m,p}$ CLIQUE**



3-SAT läßt sich auf Clique reduzieren

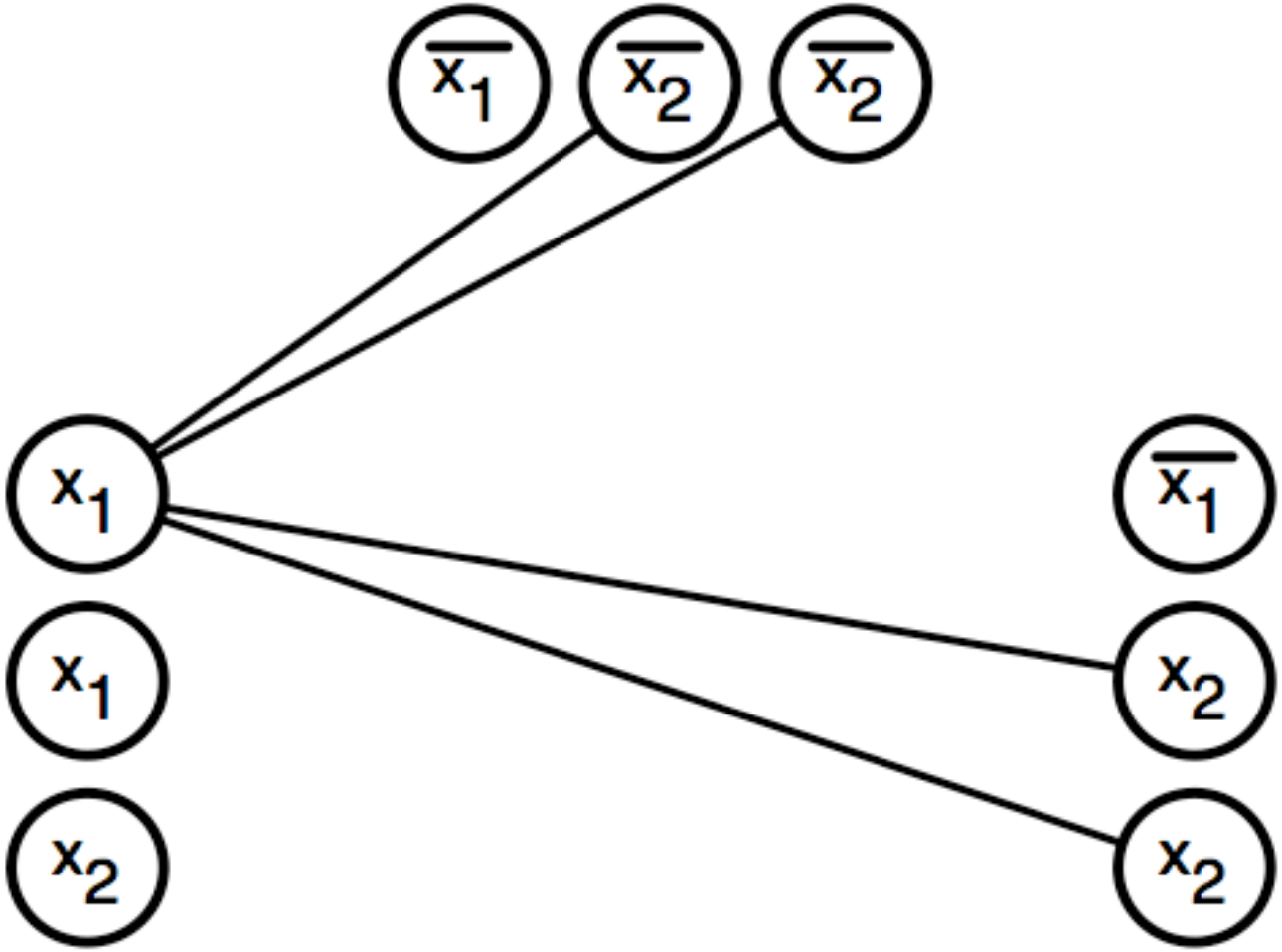
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem: 3-SAT $\leq_{m,p}$ CLIQUE**
- **Beweis**
  - Konstruiere Reduktionsfunktion f wie folgt:
  - $f(\phi) = \langle G,k \rangle$
  - k = Anzahl der Klauseln
  - Für jede Klausel C in ϕ werden drei Knoten angelegt, die mit den Literalen der Klausel bezeichnet werden
  - Füge Kante zwischen zwei Knoten ein, gdw.
    - die beiden Knoten nicht zur selben Klausel gehören und
    - die beiden Knoten nicht einer Variable und der selben negierten Variable entsprechen.

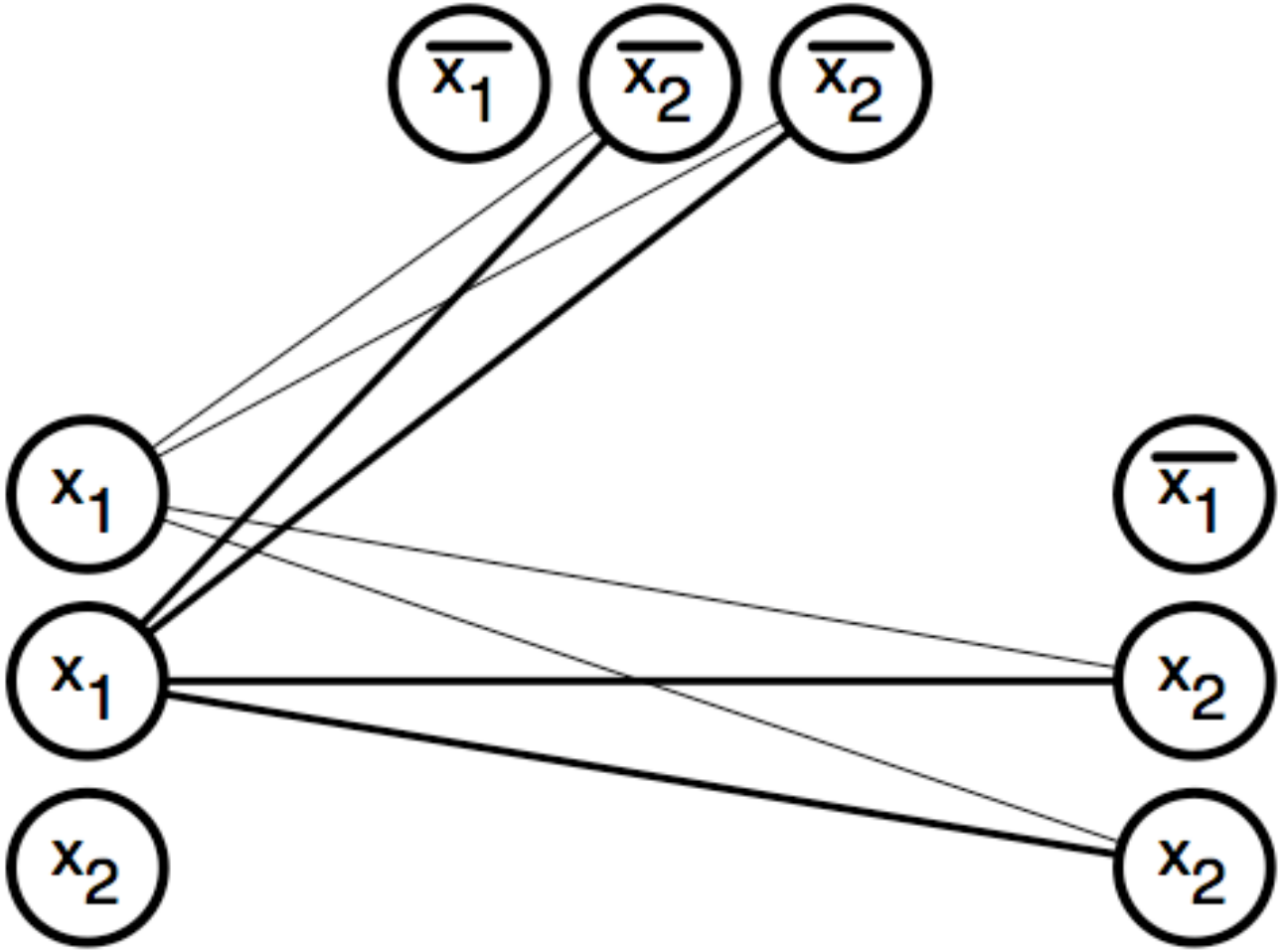


$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

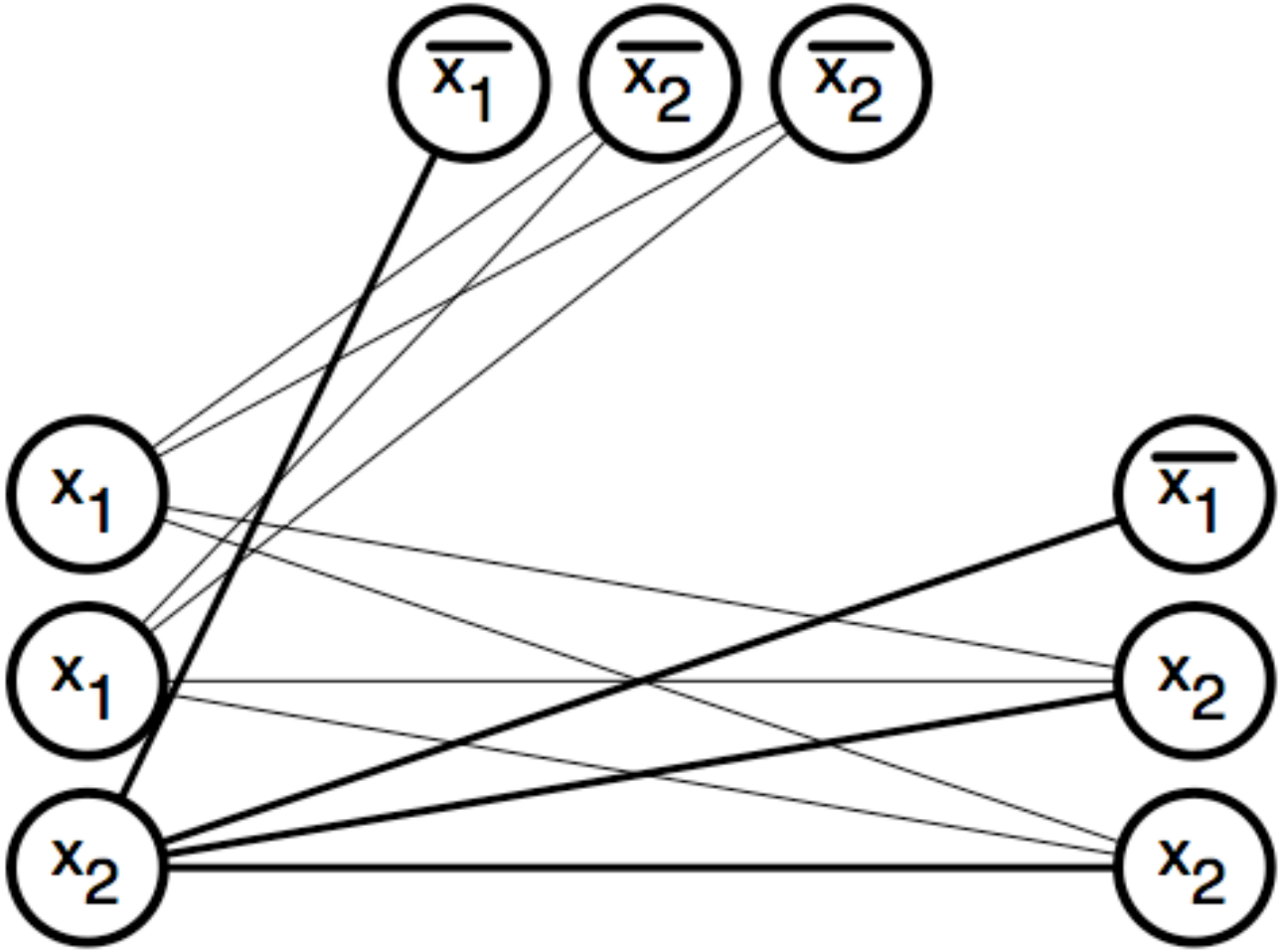
$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



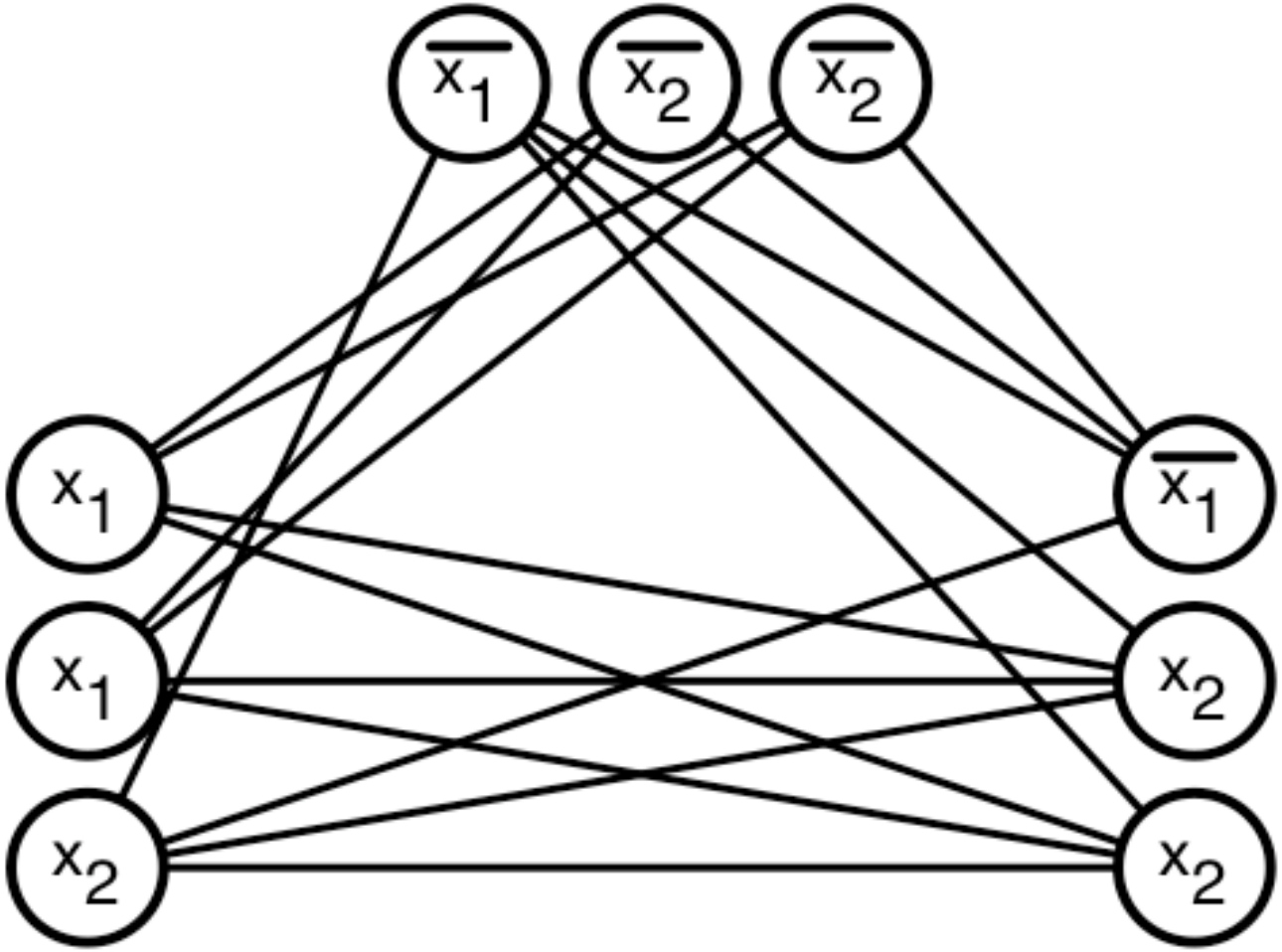
$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



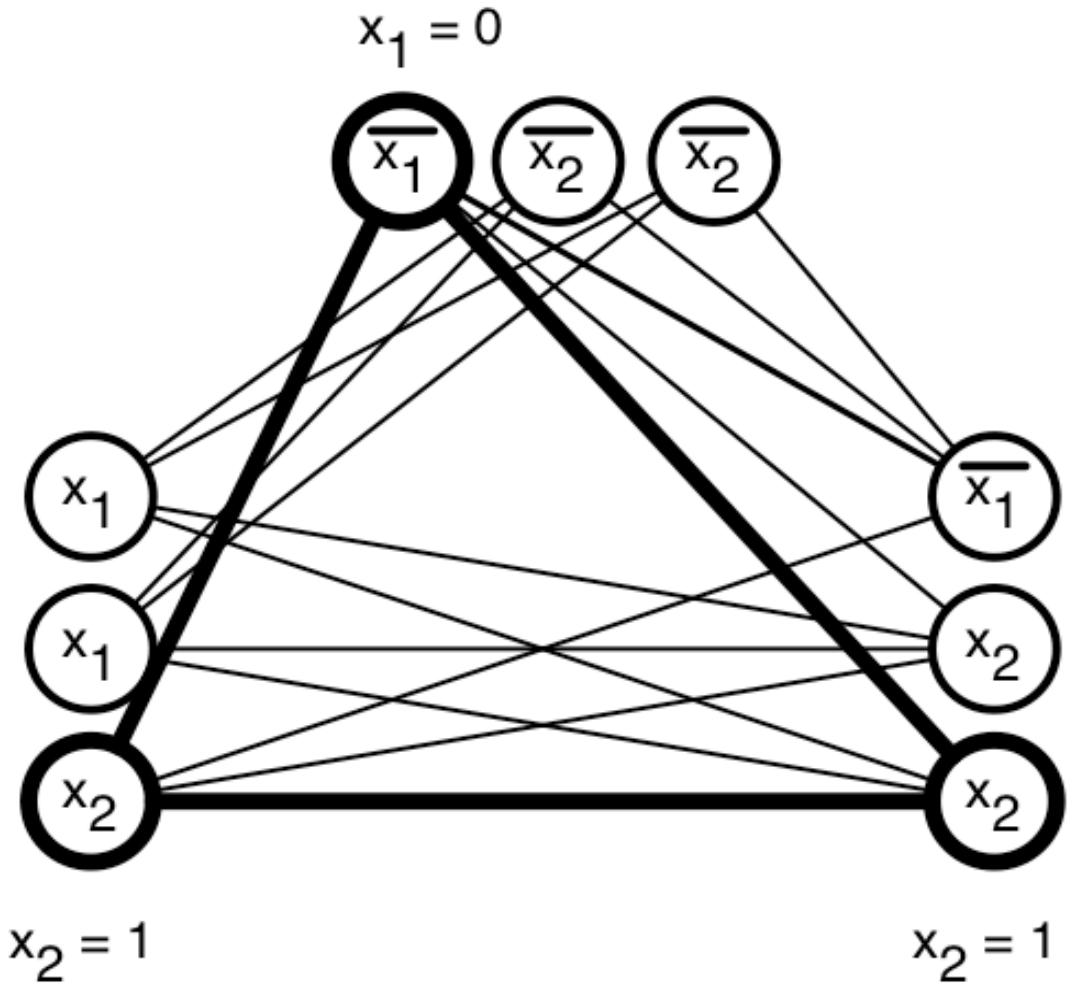
$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

0	0	**1**	**1**	0	0	**1**	**1**	**1**

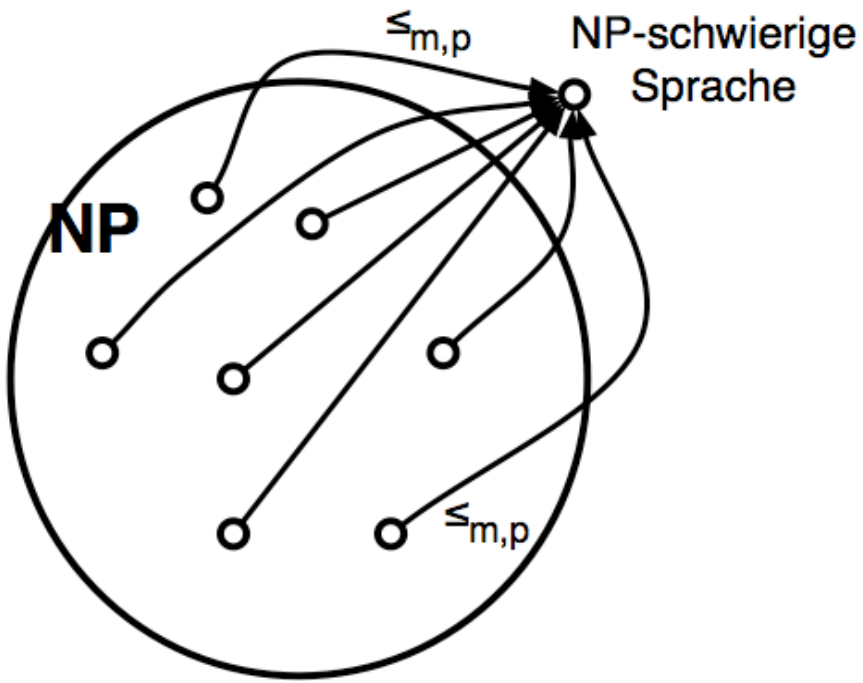
Beweis der Korrektheit der Reduktionsfunktion

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Die Reduktionsfunktion ist korrekt:**
- **Behauptung;**
  - Eine erfüllende Belegung in ϕ existert gdw. eine k-Clique in G existiert
- **1. Fall: eine erfüllende Belegung existiert in ϕ**
  - Dann liefert die Belegung in jeder Klausel mindestens ein Literal mit Wert 1
  - Wähle aus der Knotenmenge einer Klausel ein beliebiges solches Literal
  - Die gewählte Knotenmenge besteht dann aus k Knoten
  - Zwischen allen Knoten existiert eine Kante, da Variable und negierte Variable nicht gleichzeitig 1 sein können
- **2. Fall: eine k-Clique existiert in G**
  - Jeder der Knoten der Clique gehört zu einer anderen Klausel
  - Setze die entsprechenden Literale auf 1
  - Bestimmte daraus die Variablen-Belegung
  - Das führt zu keinem Widerspruch, da keine Kanten zwischen einem Literal und seiner negierten Version existieren
- **Laufzeit:**
  - Konstruktion des Graphens und der Kanten benötigt höchstens quadratische Zeit.

NP-Schwierig

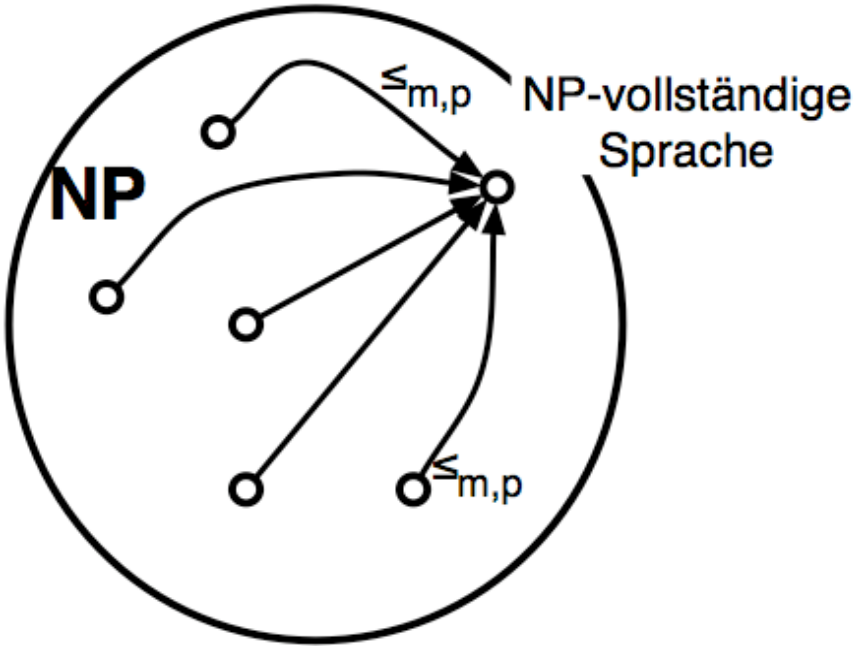
- **Definition:**
  - Eine Sprache S ist **NP-schwierig** (NP-hard) wenn:
    - jede Sprache aus NP mit einer Polynom-Zeit-Abbildungsreduktion auf S reduziert werden kann, d.h.
    - für alle $L \in NP$: $L \leq_{m,p} S$
- **Theorem**
  - Falls eine NP-schwierige Sprache in P ist, ist P=NP
- **Beweis**
  - Falls $S \in P$ und $L \leq_{m,p} S$ gilt $L \in P$.



NP-Vollständigkeit

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

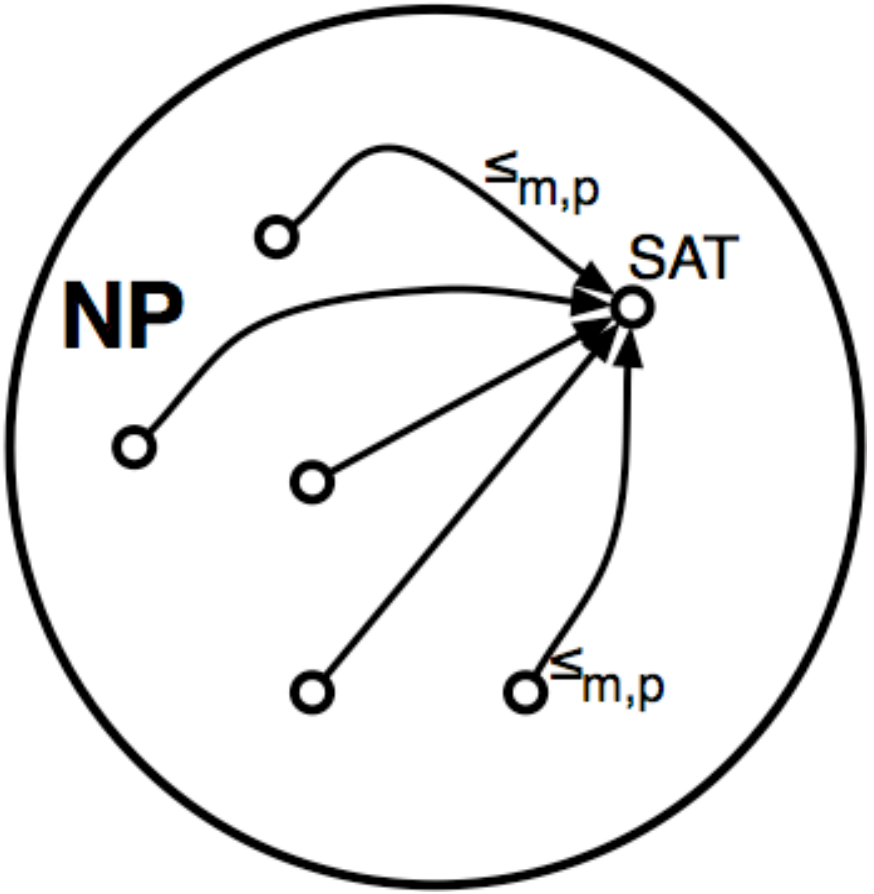
- **Definition:**
  - Eine Sprache S ist **NP-vollständig** (NP-complete) wenn:
    - S ∈ NP
    - S ist NP-schwierig
- **Korollar:**
  - Ist eine NP-vollständige Sprache in P, dann ist P=NP
- **Beweis:**
  - folgt aus der NP-Schwierigkeit der NP-vollständigen Sprache.



Der Satz von Cook und Levin

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem (Cook/Levin)**
  - SAT ist NP-vollständig



Konsequenzen aus dem Satz von Cook und Levin

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **1. Fall: SAT $\in$ P:**
  - SAT ist NP-vollständig:
    - Für alle L $\in$ NP: $L \leq_{m,p} SAT$
  - Daraus folgt für alle L $\in$ NP:
    - L $\in$ P
  - Damit ist **P=NP**
- **2. Fall: SAT $\notin$ P:**
  - Damit existiert eine Sprache in NP, die nicht in P ist
  - Damit ist **P$\neq$NP**

- **Also folgt aus dem Satz von Cook und Levin:**
  - **SAT $\in$ P $\Leftrightarrow$ P = NP**

Ende der 20. Vorlesung

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Christian Schindelhauer

Wintersemester 2006/07

20. Vorlesung

12.01.2007