Theoretische Grundlagen der Informatik Übung

5. Übungstermin · 12. Dezember

Torsten Ueckerdt

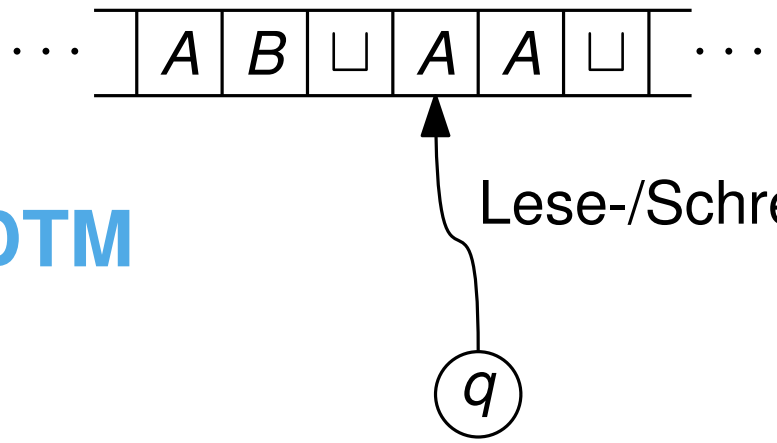
INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Übersicht

Inhalt

- Nichtdeterministische Turingmaschine(n)
- Polynomielle Reduktion
- $\mathcal{NP}$-vollständige Probleme

Nichtdeterministische Turingmaschine(n)



DTM

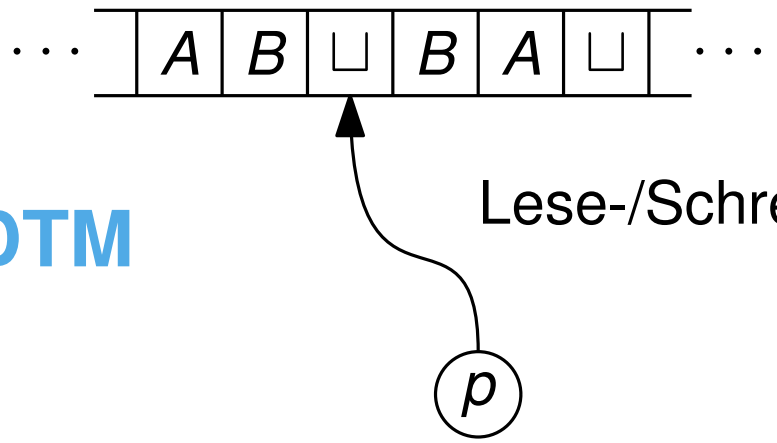
Deterministische Turing-Maschine

Übergangsfuntion

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$$

z.B. $\delta(q, A) = (p, B, L)$

Nichtdeterministische Turingmaschine(n)



Eingabe- und Rechenband

Lese-/Schreibkopf

endliche Kontrolle

DTM

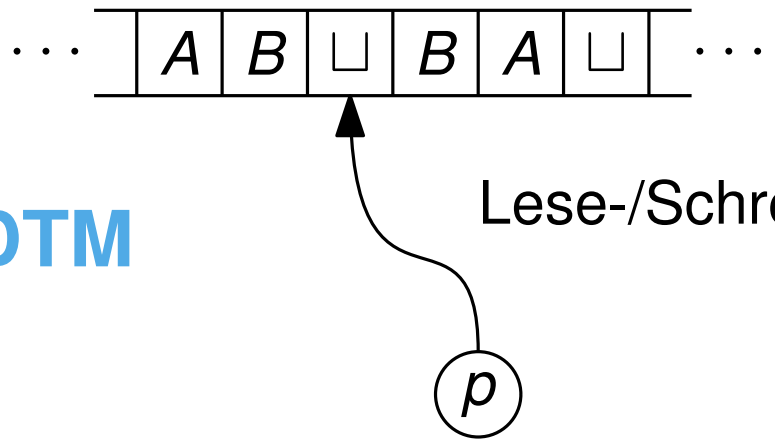
Deterministische Turing-Maschine

Übergangsfuntion

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$

z.B. $\delta(q, A) = (p, B, L)$

Nichtdeterministische Turingmaschine(n)



Eingabe- und Rechenband

DTM

Lese-/Schreibkopf

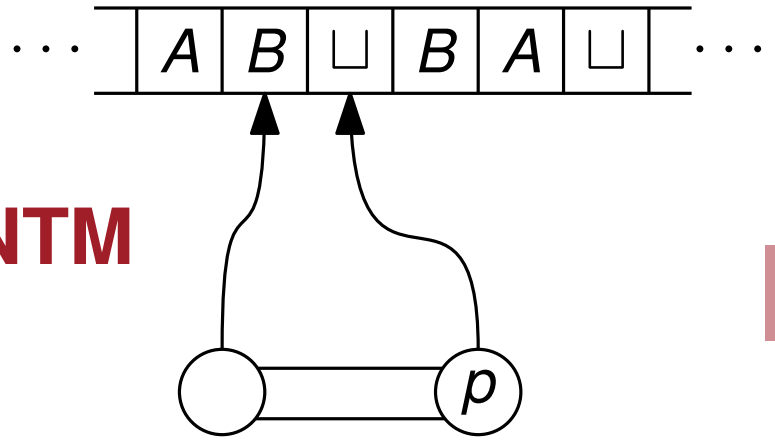
endliche Kontrolle

Deterministische Turing-Maschine

Übergangsfuntion

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$

z.B. $\delta(q, A) = (p, B, L)$



NTM

Orakel

Nichtdeterministische Turing-Maschine

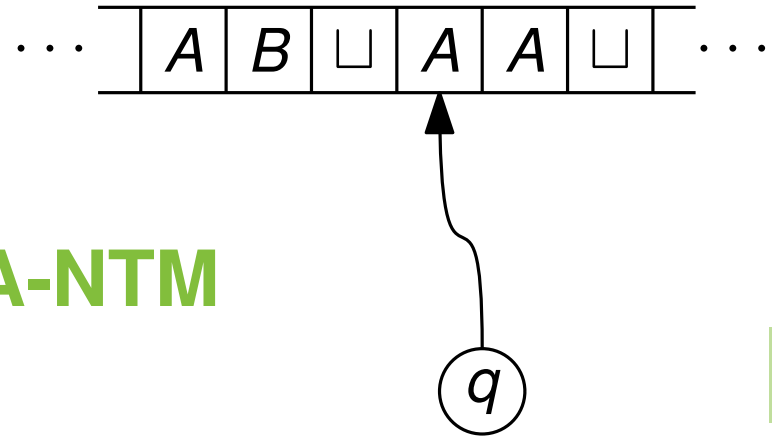
Orakelmodul

(1) Orakel schreibt Wort vor die Eingabe

(2) Orakel wird inaktiv

(3) endliche Kontrolle wird aktiv

Nichtdeterministische Turingmaschine(n)



A-NTM

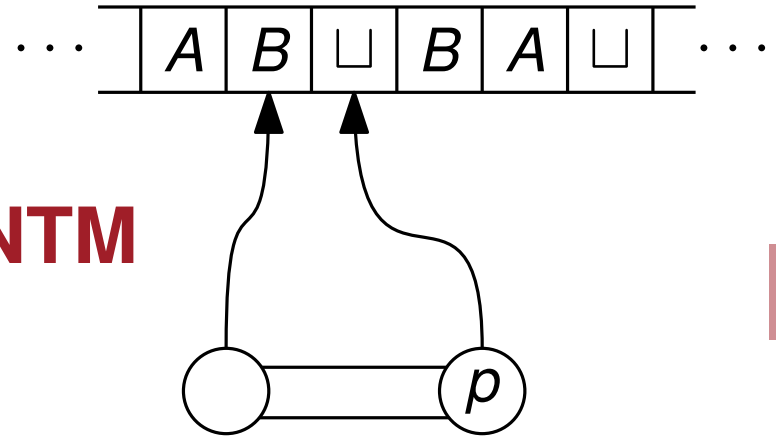
Wahlmöglichkeiten

Alternative Nichtdeterministische Turing-Maschine

Erweiterte Übergangsfuntion

$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,N,R\}}$

z.B. $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$



NTM

Orakel

Nichtdeterministische Turing-Maschine

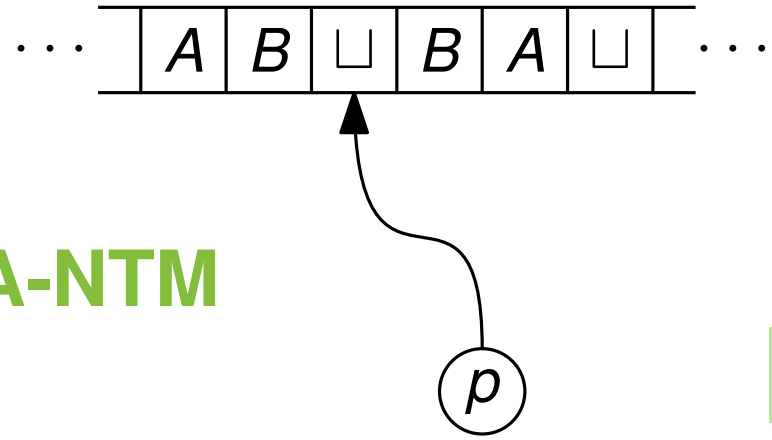
Orakelmodul

(1) Orakel schreibt Wort vor die Eingabe

(2) Orakel wird inaktiv

(3) endliche Kontrolle wird aktiv

Nichtdeterministische Turingmaschine(n)



A-NTM

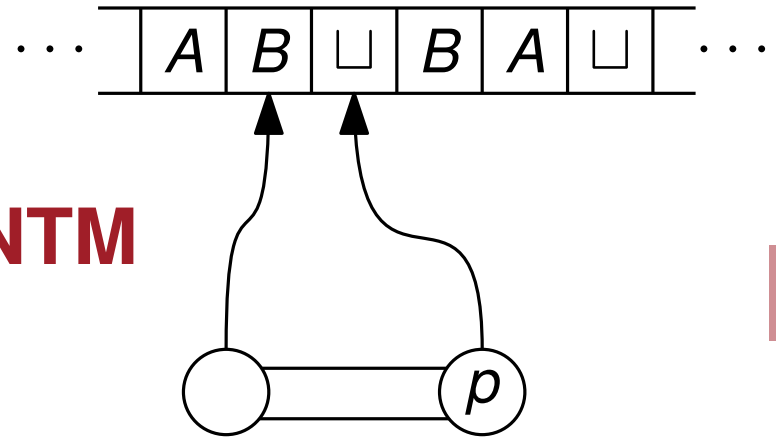
Wahlmöglichkeiten

Alternative Nichtdeterministische Turing-Maschine

Erweiterte Übergangsfuntion

$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,N,R\}}$

z.B. $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$



NTM

Orakel

Nichtdeterministische Turing-Maschine

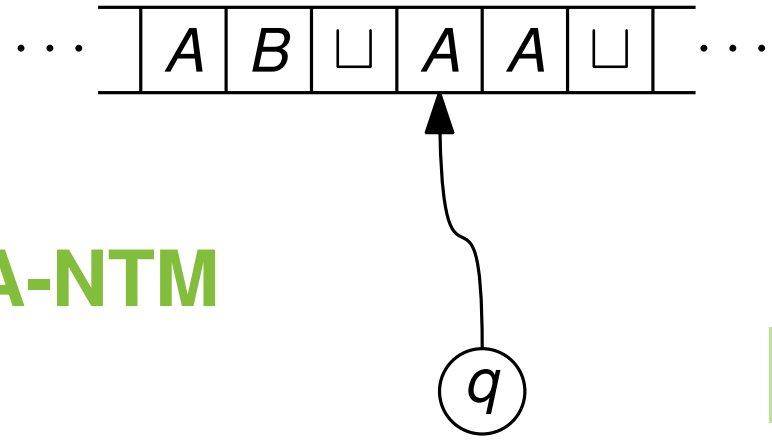
Orakelmodul

(1) Orakel schreibt Wort vor die Eingabe

(2) Orakel wird inaktiv

(3) endliche Kontrolle wird aktiv

Nichtdeterministische Turingmaschine(n)



A-NTM

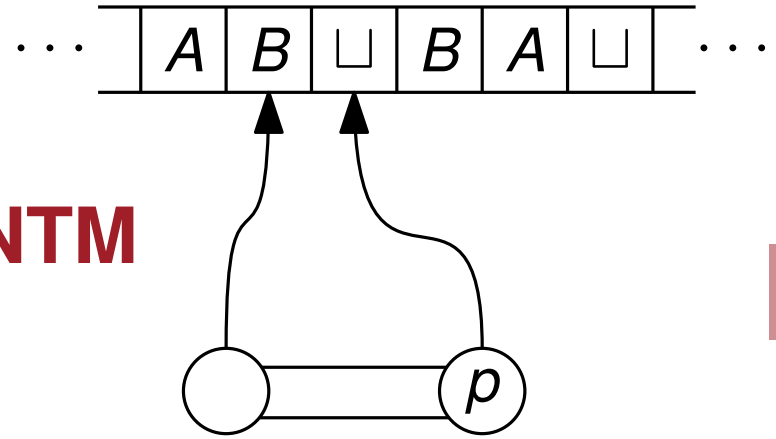
Wahlmöglichkeiten

Alternative Nichtdeterministische Turing-Maschine

Erweiterte Übergangsfuntion

$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,N,R\}}$

z.B. $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$



NTM

Orakel

Nichtdeterministische Turing-Maschine

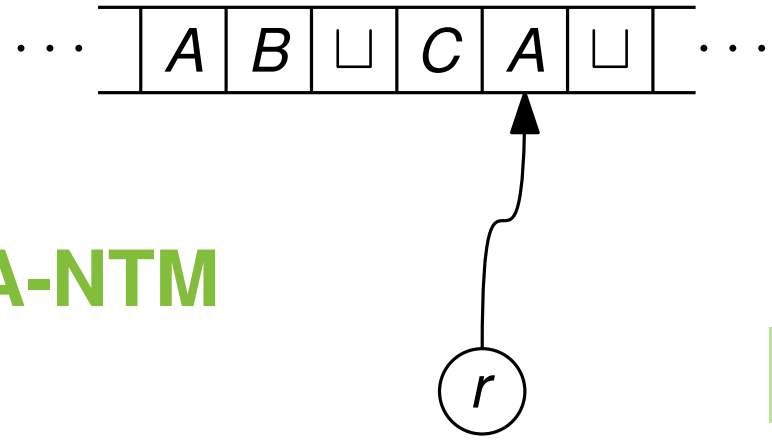
Orakelmodul

(1) Orakel schreibt Wort vor die Eingabe

(2) Orakel wird inaktiv

(3) endliche Kontrolle wird aktiv

Nichtdeterministische Turingmaschine(n)



A-NTM

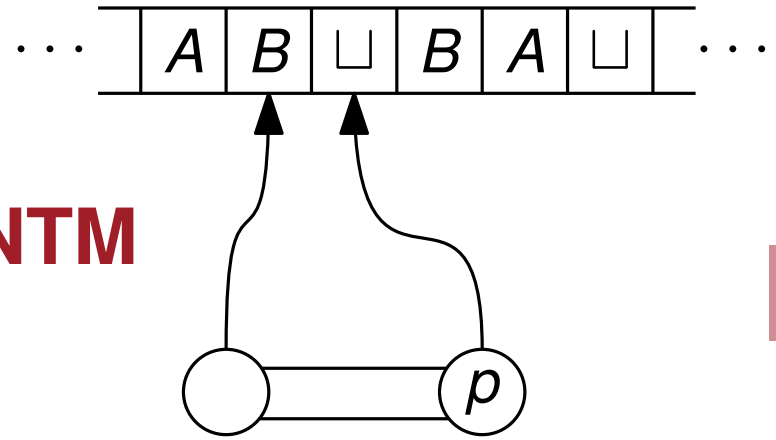
Wahlmöglichkeiten

Alternative Nichtdeterministische Turing-Maschine

Erweiterte Übergangsfuntion

$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,N,R\}}$

z.B. $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$



NTM

Orakel

Nichtdeterministische Turing-Maschine

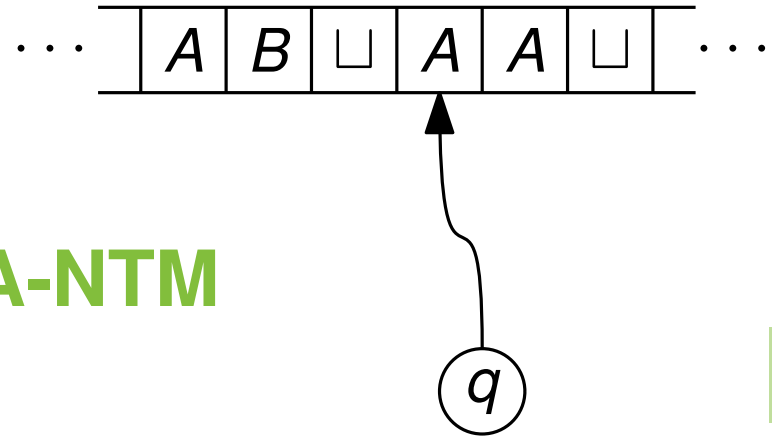
Orakelmodul

(1) Orakel schreibt Wort vor die Eingabe

(2) Orakel wird inaktiv

(3) endliche Kontrolle wird aktiv

Nichtdeterministische Turingmaschine(n)



A-NTM

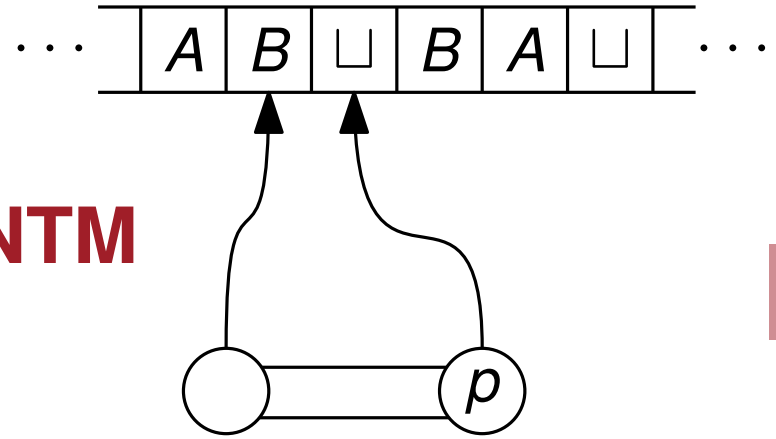
Wahlmöglichkeiten

Alternative Nichtdeterministische Turing-Maschine

Erweiterte Übergangsfuntion

$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,N,R\}}$

z.B. $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$



NTM

Orakel

Nichtdeterministische Turing-Maschine

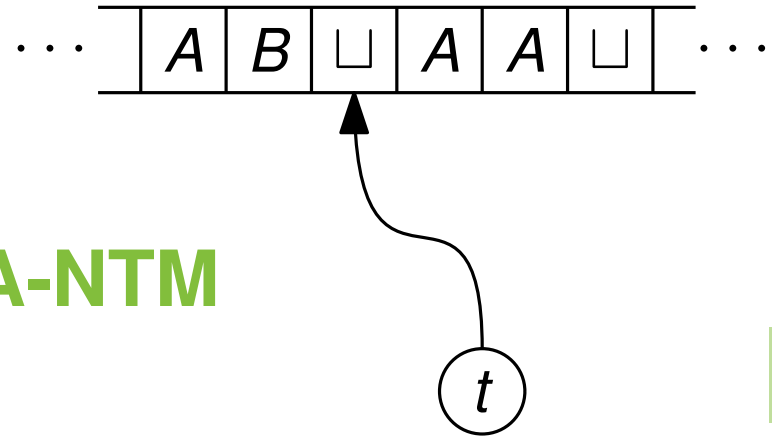
Orakelmodul

(1) Orakel schreibt Wort vor die Eingabe

(2) Orakel wird inaktiv

(3) endliche Kontrolle wird aktiv

Nichtdeterministische Turingmaschine(n)



A-NTM

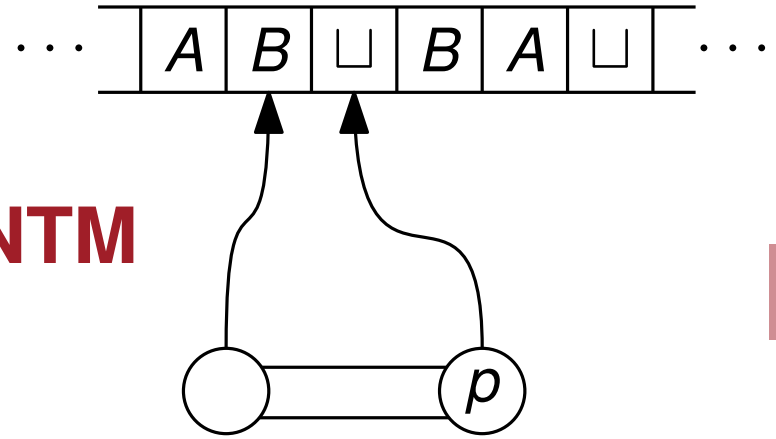
Wahlmöglichkeiten

Alternative Nichtdeterministische Turing-Maschine

Erweiterte Übergangsfuntion

$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,N,R\}}$

z.B. $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$



NTM

Orakel

Nichtdeterministische Turing-Maschine

Orakelmodul

(1) Orakel schreibt Wort vor die Eingabe

(2) Orakel wird inaktiv

(3) endliche Kontrolle wird aktiv

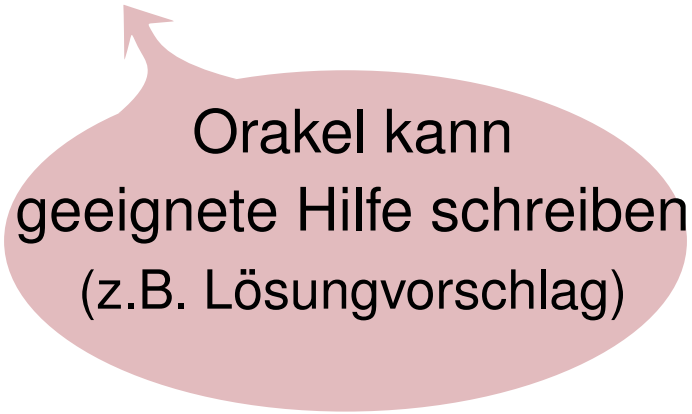
Die Klassen $\mathcal{NP}$ und $\mathcal{ANP}$

Eine **NTM** akzeptiert ein Wort x wenn es eine Möglichkeit gibt, bei Eingabe x in q_J zu enden.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Die Klassen $\mathcal{NP}$ und $\mathcal{ANP}$

Eine **NTM** akzeptiert ein Wort x wenn es eine Möglichkeit gibt, bei Eingabe x in q_J zu enden.



Die Klassen $\mathcal{NP}$ und $\mathcal{ANP}$

Eine **NTM** akzeptiert ein Wort x wenn es eine Möglichkeit gibt,
bei Eingabe x in q_J zu enden.

Die Klasse $\mathcal{NP}$:
= alle Sprachen, die von einer **NTM**
in polynomialer Laufzeit akzeptiert werden

Orakel kann
geeignete Hilfe schreiben
(z.B. Lösungvorschlag)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Die Klassen $\mathcal{NP}$ und $\mathcal{ANP}$

Eine **NTM** akzeptiert ein Wort x wenn es eine Möglichkeit gibt, bei Eingabe x in q_J zu enden.

Die Klasse $\mathcal{NP}$:
= alle Sprachen, die von einer **NTM** in polynomialer Laufzeit akzeptiert werden

Orakel kann geeignete Hilfe schreiben (z.B. Lösungvorschlag)

Eine **A-NTM** akzeptiert ein Wort x wenn es eine Möglichkeit gibt, bei Eingabe x in q_J zu enden.

Übergangsfunktion kann die richtigen Entscheidungen treffen

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Die Klassen $\mathcal{NP}$ und $\mathcal{ANP}$

Eine **NTM** akzeptiert ein Wort x wenn es eine Möglichkeit gibt, bei Eingabe x in q_J zu enden.

Die Klasse $\mathcal{NP}$:
= alle Sprachen, die von einer **NTM** in polynomialer Laufzeit akzeptiert werden

Orakel kann geeignete Hilfe schreiben (z.B. Lösungvorschlag)

Eine **A-NTM** akzeptiert ein Wort x wenn es eine Möglichkeit gibt, bei Eingabe x in q_J zu enden.

Die Klasse $\mathcal{ANP}$:
= alle Sprachen, die von einer **A-NTM** in polynomialer Laufzeit akzeptiert werden

Übergangsfunktion kann die richtigen Entscheidungen treffen

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Wir müssen zeigen:

Zu jeder **A-NTM** M gibt es eine **NTM** M', so dass

1. M und M' akzeptieren die gleiche Sprache
2. Laufzeit von M' ist beschränkt durch ein Polynom in der Laufzeit von M

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Wir müssen zeigen:

Zu jeder **A-NTM** M gibt es eine **NTM** M', so dass

1. M und M' akzeptieren die gleiche Sprache
2. Laufzeit von M' ist beschränkt durch ein Polynom in der Laufzeit von M

Konstruktion von **NTM** M' zu gegebener **A-NTM** M:

- Sei $|\delta_M(q, A)| \leq K$ für alle (q, A). (max. # Wahlmöglichkeiten in M)
- Nummeriere Optionen in $\delta_M(q, A)$ von 1 bis max. K.
- O.B.d.A. seien $1, \ldots, K \notin \Gamma_M$.
- **Idee:** Orakel von M' schreibt Zahlenfolge der "richtigen Entscheidungen"

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Konstruktion von **NTM** M' zu gegebener **A-NTM** M:

- Sei $|\delta_M(q, A)| \leq K$ für alle (q, A). (max. # Wahlmöglichkeiten in M)
- Nummeriere Optionen in $\delta_M(q, A)$ von 1 bis max. K.
- O.B.d.A. seien $1, \ldots, K \notin \Gamma_M$.
- **Idee:** Orakel von M' schreibt Folge der "richtigen Entscheidungen"
- Definiere deterministischen Teil von M' als **2-Kopf TM** M'_{det} durch

$\delta_{M'_{\text{det}}}(q, (i, A)) = (p, (i, B), (R, D))$ gdw. i-te Option in $\delta_M(q, A)$ ist (p, B, D)

2. Kopf, endl. Kontrolle folgen $\delta_M(q, A)$

1. Kopf liest Orakel-Vorgabe

- 2-Kopf TM kann durch 1-Kopf TM simuliert werden (siehe Übung 3)

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Wir müssen zeigen:

Zu jeder **A-NTM** M gibt es eine **NTM** M', so dass

1. M und M' akzeptieren die gleiche Sprache
2. Laufzeit von M' ist beschränkt durch ein Polynom in der Laufzeit von M

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Wir müssen zeigen:

Zu jeder **A-NTM** M gibt es eine **NTM** M', so dass

1. M und M' akzeptieren die gleiche Sprache
2. Laufzeit von M' ist beschränkt durch ein Polynom in der Laufzeit von M

M akzeptiert Eingabe x.

$\Rightarrow$ Es gibt Übergänge*, die M bei Eingabe x in q_J überführen.

$\Rightarrow$ Es gibt Orakel-Hinweis der M' bei Eingabe x in q_J überführt.

$\Rightarrow$ M' akzeptiert Eingabe x.

* aka. Entscheidungen bei Wahlmöglichkeiten

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Wir müssen zeigen:

Zu jeder **A-NTM** M gibt es eine **NTM** M', so dass

1. M und M' akzeptieren die gleiche Sprache
2. Laufzeit von M' ist beschränkt durch ein Polynom in der Laufzeit von M

M akzeptiert Eingabe x.

$\Longleftrightarrow$ Es gibt Übergänge*, die M bei Eingabe x in q_J überführen.

$\Longleftrightarrow$ Es gibt Orakel-Hinweis der M' bei Eingabe x in q_J überführt.

$\Longleftrightarrow$ M' akzeptiert Eingabe x.

* aka. Entscheidungen bei Wahlmöglichkeiten

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Wir müssen zeigen:

Zu jeder **A-NTM** M gibt es eine **NTM** M', so dass

1. M und M' akzeptieren die gleiche Sprache
2. Laufzeit von M' ist beschränkt durch ein Polynom in der Laufzeit von M

Laufzeit von M' bei akzeptierter Eingabe x

$\leq$ Orakel-Schreibezeit + #Schritte, um M'_{det} in q_J zu überführen

$\leq 2 \cdot$ #Schritte, um M in q_J zu überführen

$\leq 2 \cdot$ Laufzeit von M

A-NTMs sind nicht mächtiger als NTMs

Behauptung: $\mathcal{ANP} \subseteq \mathcal{NP}$

Wir müssen zeigen:

Zu jeder **A-NTM** M gibt es eine **NTM** M', so dass

1. M und M' akzeptieren die gleiche Sprache
2. Laufzeit von M' ist beschränkt durch ein Polynom in der Laufzeit von M

Laufzeit von M' bei akzeptierter Eingabe x

$\leq$ Orakel-Schreibezeit + #Schritte, um M'_{det} in q_J zu überführen

$\leq 2 \cdot$ #Schritte, um M in q_J zu überführen

$\leq 2 \cdot$ Laufzeit von M

Damit ist die Behauptung $\mathcal{ANP} \subseteq \mathcal{NP}$ gezeigt.

Übersicht

Inhalt

- Nichtdeterministische Turingmaschine(n)
- Polynomielle Reduktion
- $\mathcal{NP}$-vollständige Probleme

Polynomiale Transformation

Eine **polynomiale Transformation** einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

- es existiert eine polynomiale deterministische Turing-Maschine, die f berechnet;
- für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.

Wir schreiben dann $L_1 \propto L_2$ (L_1 ist polynomial transformierbar in L_2).

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Polynomielle Transformation

L_1 = Sprache der ungeraden Zahlen in Dezimaldarstellung über $\Sigma_1 = \{0, \ldots, 9\}$

L_2 = Sprache der Wörter gerader Länge über $\Sigma_2 = \{a, b\}$

Zeigen Sie, dass L_1 polynomial in L_2 transformiert werden kann ($L_1 \propto L_2$).

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Polynomielle Transformation

L_1 = Sprache der ungeraden Zahlen in Dezimaldarstellung über $\Sigma_1 = \{0, \ldots, 9\}$

L_2 = Sprache der Wörter gerader Länge über $\Sigma_2 = \{a, b\}$

Zeigen Sie, dass L_1 polynomial in L_2 transformiert werden kann ($L_1 \propto L_2$).

Sei $f : \Sigma_1^* \to \Sigma_2^*$ definiert als

$$f(z) = \begin{cases} aa & \text{falls } z \text{ ungerade ist} \\ a & \text{sonst} \end{cases}$$

Beispiel:

$f(1301) = aa$

$f(1000) = a$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Polynomielle Transformation

L_1 = Sprache der ungeraden Zahlen in Dezimaldarstellung über $\Sigma_1 = \{0, \ldots, 9\}$

L_2 = Sprache der Wörter gerader Länge über $\Sigma_2 = \{a, b\}$

Zeigen Sie, dass L_1 polynomial in L_2 transformiert werden kann ($L_1 \propto L_2$).

Sei $f : \Sigma_1^* \to \Sigma_2^*$ definiert als

$$f(z) = \begin{cases} aa & \text{falls } z \text{ ungerade ist} \\ a & \text{sonst} \end{cases}$$

Beispiel:

$f(1301) = aa$

$f(1000) = a$

1 Schritt: Zeige, dass f von DTM berechnet werden kann.

- DTM erhält als Eingabe eine Dezimalzahl.
- DTM löscht alle Zeichen, bis auf das letzte Zeichen.
- Wenn verbleibendes Zeichen ungerade, dann überschreibe mit aa
- **Ansonsten** überschreibe mit a
- Polynomieller Zeitaufwand.

Polynomielle Transformation

L_1 = Sprache der ungeraden Zahlen in Dezimaldarstellung über $\Sigma_1 = \{0, \ldots, 9\}$

L_2 = Sprache der Wörter gerader Länge über $\Sigma_2 = \{a, b\}$

Zeigen Sie, dass L_1 polynomial in L_2 transformiert werden kann ($L_1 \propto L_2$).

Sei $f : \Sigma_1^* \to \Sigma_2^*$ definiert als

$$f(z) = \begin{cases} aa & \text{falls } z \text{ ungerade ist} \\ a & \text{sonst} \end{cases}$$

Beispiel:

$f(1301) = aa$

$f(1000) = a$

2. Schritt: Zeige, dass für jedes $w \in \Sigma_1^\star$ gilt: $w \in L_1 \Leftrightarrow f(w) \in L_2$

$w \in L_1 \Leftrightarrow w$ ist ungerade. $\Leftrightarrow f(w) = aa \Leftrightarrow f(w) \in L_2$.

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:

= alle Probleme, die von einer **NTM** in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:

= alle Probleme, die von einer **DTM** in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:

= alle Probleme Π, so dass für alle Probleme in $\Pi' \in \mathcal{NP}$ Π' polynomial reduzierbar auf Π d.h. $\Pi' \propto \Pi$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM**
in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM**
in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass
für alle Probleme in $\Pi' \in \mathcal{NP}$
Π' polynomial reduzierbar auf Π
d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM**
in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM**
in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass
für alle Probleme in $\Pi' \in \mathcal{NP}$
Π' polynomial reduzierbar auf Π
d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Ist $\Pi \in \mathcal{NP}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{NP}$.

(2) Konstruiere **NTM** die Π löst.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM** in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM** in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass für alle Probleme in $\Pi' \in \mathcal{NP}$ Π' polynomial reduzierbar auf Π d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Ist $\Pi \in \mathcal{NP}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{NP}$.

(2) Konstruiere **NTM** die Π löst.

Ist Π $\mathcal{NP}$-schwer?

(1) Zeige $\Pi' \propto \Pi$ für bekanntes Π' $\mathcal{NP}$-schwer.

(2) Beweise ad hoc, dass Π $\mathcal{NP}$-schwer.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM** in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM** in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass für alle Probleme in $\Pi' \in \mathcal{NP}$ Π' polynomial reduzierbar auf Π d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Ist $\Pi \in \mathcal{NP}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{NP}$.

(2) Konstruiere **NTM** die Π löst.

Ist Π $\mathcal{NP}$-schwer?

(1) Zeige $\Pi' \propto \Pi$ für bekanntes Π' $\mathcal{NP}$-schwer.

(2) Beweise ad hoc, dass Π $\mathcal{NP}$-schwer.

gute Idee

schlechte Idee

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM** in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM** in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass für alle Probleme in $\Pi' \in \mathcal{NP}$ Π' polynomial reduzierbar auf Π d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Ist $\Pi \in \mathcal{NP}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{NP}$.

(2) Konstruiere **NTM** die Π löst.

Ist Π $\mathcal{NP}$-schwer?

(1) Zeige $\Pi' \propto \Pi$ für bekanntes Π' $\mathcal{NP}$-schwer.

(2) Beweise ad hoc, dass Π $\mathcal{NP}$-schwer.

gute Idee

schlechte Idee

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM** in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM** in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass für alle Probleme in $\Pi' \in \mathcal{NP}$ Π' polynomial reduzierbar auf Π d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Ist $\Pi \in \mathcal{NP}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{NP}$.

(2) Konstruiere **NTM** die Π löst.

Ist Π $\mathcal{NP}$-schwer?

(1) Zeige $\Pi' \propto \Pi$ für bekanntes Π' $\mathcal{NP}$-schwer.

(2) Beweise ad hoc, dass Π $\mathcal{NP}$-schwer.

gute Idee

schlechte Idee

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

2SAT $\in \mathcal{P}$

2SAT $\in \mathcal{P}$

Problem 2SAT:

- **Gegeben:** Menge U von Variablen, Menge von C Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält.
- **Gefragt:** Existiert eine erfüllende Wahrheitsbelegung für C?

Beispiel:

$$U = \{ x_1, x_2, x_3 \}$$

$$C = \{ \neg x_1 \vee x_2, \neg x_2 \vee x_3, x_1 \vee \neg x_3, x_2 \vee x_3 \}$$

erfüllend: $x_1 = x_2 = x_3 = \text{TRUE}$

nicht erfüllend: $x_1 = \text{TRUE}, x_2 = x_3 = \text{FALSE}$

Aufgabe: Zeigen Sie, dass 2SAT in $\mathcal{P}$ liegt.

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM** in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM** in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass für alle Probleme in $\Pi' \in \mathcal{NP}$ Π' polynomial reduzierbar auf Π d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Ist $\Pi \in \mathcal{NP}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{NP}$.

(2) Konstruiere **NTM** die Π löst.

Ist Π $\mathcal{NP}$-schwer?

(1) Zeige $\Pi' \propto \Pi$ für bekanntes Π' $\mathcal{NP}$-schwer.

(2) Beweise ad hoc, dass Π $\mathcal{NP}$-schwer.

gute Idee

schlechte Idee

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

2SAT $\in \mathcal{P}$

Konstruiere gerichteten Graphen $G = (V, E)$

$V = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$

$E = \{(\neg l_1, l_2) \mid l_1 \vee l_2 \in C \text{ oder } l_2 \vee l_1 \in C\}$

Es gilt:

$a \vee b \equiv \neg a \Rightarrow b \equiv \neg b \Rightarrow a$

2SAT $\in \mathcal{P}$

Konstruiere gerichteten Graphen $G = (V, E)$

$V = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$

$E = \{(\neg l_1, l_2) \mid l_1 \vee l_2 \in C \text{ oder } l_2 \vee l_1 \in C\}$

Gebe $G = (V, E)$ für folgende zwei Instanzen an:

Instanz 1: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

Instanz 2: $x_1 \vee x_2 \quad x_1 \vee \neg x_2 \quad \neg x_1 \vee x_2 \quad \neg x_1 \vee \neg x_2$

5 min Zeit

Es gilt:

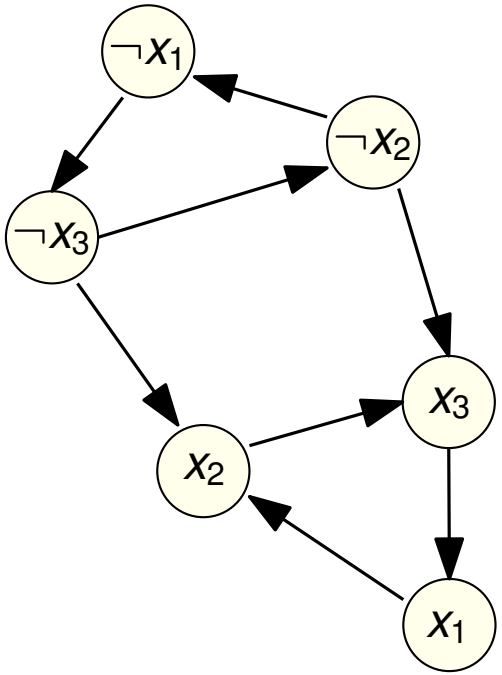
$a \vee b \equiv \neg a \Rightarrow b \equiv \neg b \Rightarrow a$

2SAT $\in \mathcal{P}$

Konstruiere gerichteten Graphen $G = (V, E)$

$V = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$

$E = \{(\neg l_1, l_2) \mid l_1 \vee l_2 \in C \text{ oder } l_2 \vee l_1 \in C\}$



Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.

Es gilt:

$a \vee b \equiv \neg a \Rightarrow b \equiv \neg b \Rightarrow a$

Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

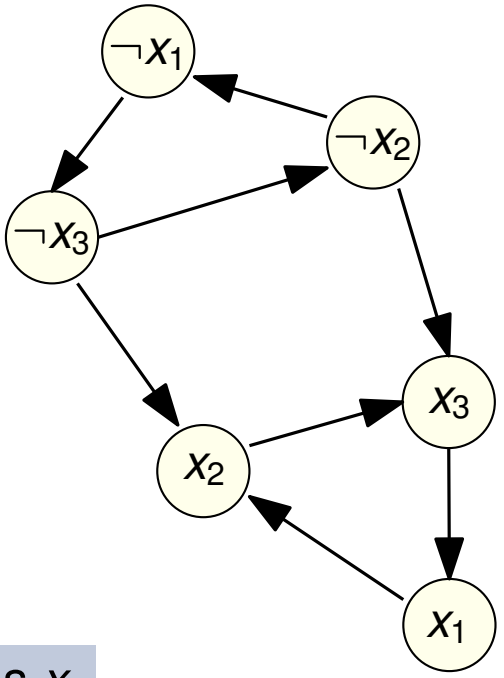
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

2SAT $\in \mathcal{P}$

Konstruiere gerichteten Graphen $G = (V, E)$

$V = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$

$E = \{(\neg l_1, l_2) \mid l_1 \vee l_2 \in C \text{ oder } l_2 \vee l_1 \in C\}$



Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.

Zeige: C ist erfüllbar gdw. es gibt keine Variable x_i, so dass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen.

Es gilt:

$a \vee b \equiv \neg a \Rightarrow b \equiv \neg b \Rightarrow a$

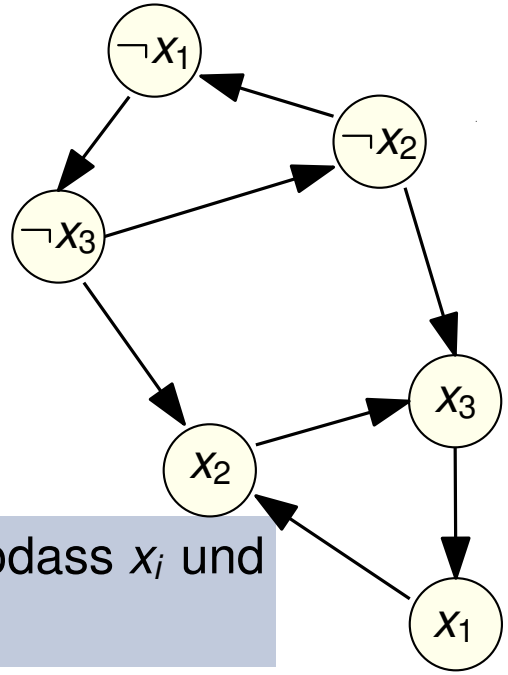
Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

2SAT $\in \mathcal{P}$

Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.



Zeige: C ist erfüllbar gdw. es gibt keine Variable x_i, sodass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen.

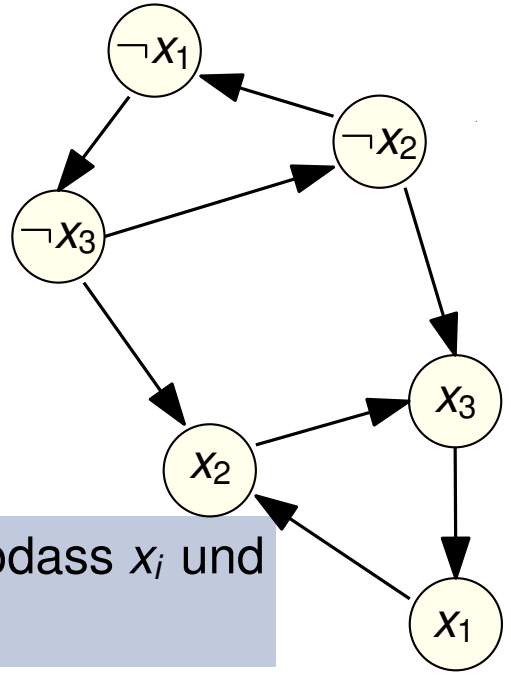
Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

2SAT $\in \mathcal{P}$

Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.

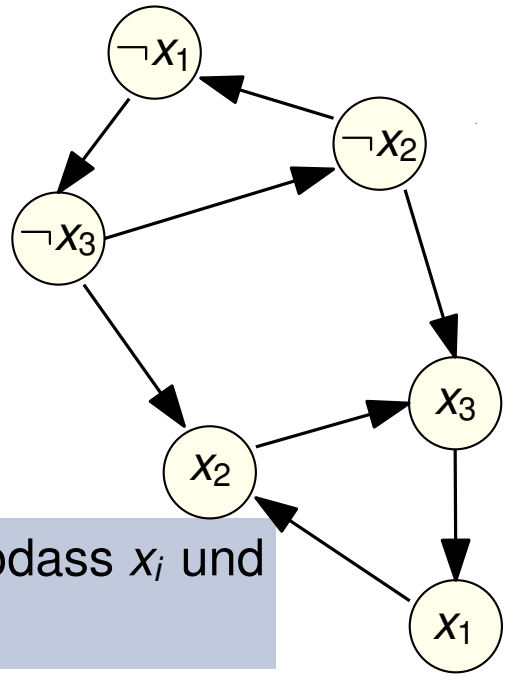


Zeige: C ist erfüllbar gdw. es gibt keine Variable x_i, sodass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen.

"$\Rightarrow$" **Annahme:** $\exists$ x_i und $\neg x_i$, die in derselben stark verbunden Komponente liegen

Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

2SAT $\in \mathcal{P}$



Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.

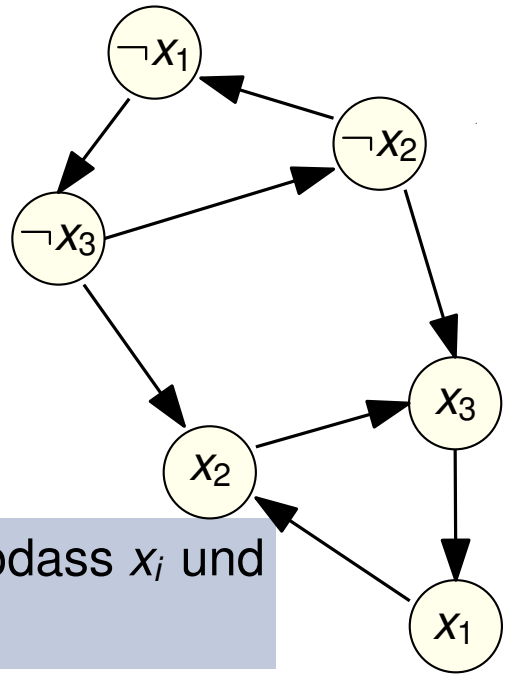
Zeige: C ist erfüllbar gdw. es gibt keine Variable x_i, sodass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen.

"$\Rightarrow$" **Annahme:** $\exists$ x_i und $\neg x_i$, die in derselben stark verbunden Komponente liegen

1. Es gibt gerichteten Pfad $x_i, \ldots, \neg x_i$. → Es gilt Implikation $x_i \Rightarrow \neg x_i$
2. Es gibt gerichteten Pfad $\neg x_i, \ldots, x_i$. → Es gilt Implikation $\neg x_i \Rightarrow x_i$

Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

2SAT $\in \mathcal{P}$



Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.

Zeige: C ist erfüllbar gdw. es gibt keine Variable x_i, sodass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen.

"$\Rightarrow$" **Annahme:** $\exists$ x_i und $\neg x_i$, die in derselben stark verbunden Komponente liegen

1. Es gibt gerichteten Pfad $x_i, \ldots, \neg x_i$. → Es gilt Implikation $x_i \Rightarrow \neg x_i$
2. Es gibt gerichteten Pfad $\neg x_i, \ldots, x_i$. → Es gilt Implikation $\neg x_i \Rightarrow x_i$

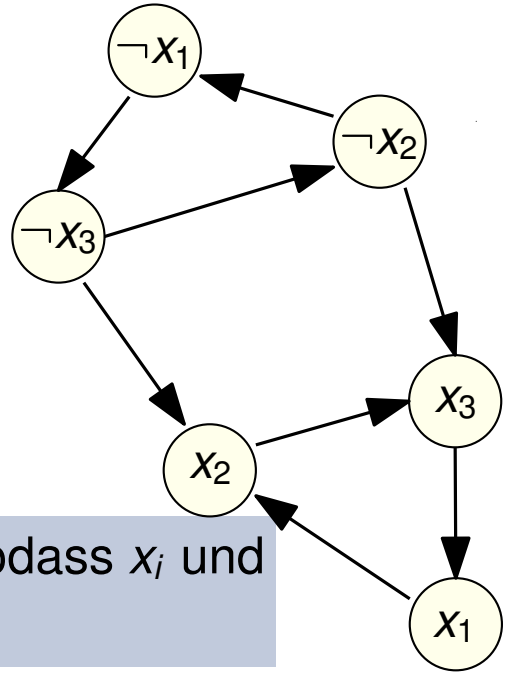
↯ C erfüllbar

Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

2SAT $\in \mathcal{P}$

Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.



Zeige: C ist erfüllbar gdw. es gibt keine Variable x_i, sodass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen.

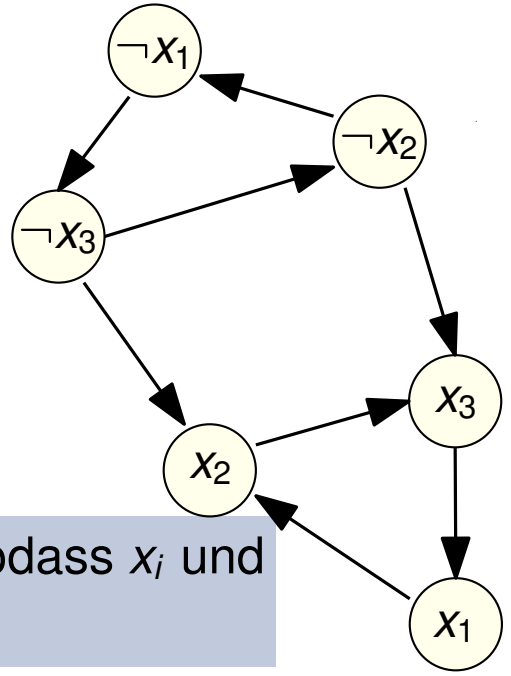
"$\Leftarrow$" Es gibt kein x_i, so dass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen

Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

2SAT $\in \mathcal{P}$

Definition: Zwei Knoten $u, v \in V$ liegen in derselben *stark verbundenen Komponente* von G, wenn

1. Es gibt gerichteten Pfad $u, \ldots, v$.
2. Es gibt gerichteten Pfad $v, \ldots, u$.



Zeige: C ist erfüllbar gdw. es gibt keine Variable x_i, sodass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen.

"$\Leftarrow$" Es gibt kein x_i, so dass x_i und $\neg x_i$ in derselben stark verbunden Komponente liegen

Setze x_i = *false*, falls es gerichteten Pfad $x_i \ldots \neg x_i$ gibt, denn es gilt $x_i \Rightarrow \neg x_i$

Setze x_i = *true*, falls es gerichteten Pfad $\neg x_i \ldots x_i$ gibt, denn es gilt $\neg x_i \Rightarrow x_i$

Beispiel: $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$\mathcal{NP}$-Vollständigkeit

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L heißt **$\mathcal{NP}$–vollständig**, falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

Aussage aus Vorlesung:

Falls $L_1, L_2 \in$ NP, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Schema für Beweis, dass Problem Π $\mathcal{NP}$-vollständig ist.

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L heißt **$\mathcal{NP}$–vollständig**, falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

Aussage aus Vorlesung:

Falls $L_1, L_2 \in$ NP, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Schema für Beweis, dass Problem Π $\mathcal{NP}$-vollständig ist.

1. Schritt: Zeige, dass Π in $\mathcal{NP}$ liegt.

2. Schritt: Zeige, dass es $\mathcal{NP}$-vollständiges Problem Π' gibt mit $\Pi' \propto \Pi$

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L heißt **$\mathcal{NP}$–vollständig**, falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

Aussage aus Vorlesung:

Falls $L_1, L_2 \in$ NP, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Schema für Beweis, dass Problem Π $\mathcal{NP}$-vollständig ist.

1. Schritt: Zeige, dass Π in $\mathcal{NP}$ liegt.

a) Orakel rät mögliche Lösung $\mathcal{L}$ für gegebene Instanz

2. Schritt: Zeige, dass es $\mathcal{NP}$-vollständiges Problem Π' gibt mit $\Pi' \propto \Pi$

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L heißt **$\mathcal{NP}$–vollständig**, falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

Aussage aus Vorlesung:

Falls $L_1, L_2 \in$ NP, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Schema für Beweis, dass Problem Π $\mathcal{NP}$-vollständig ist.

1. Schritt: Zeige, dass Π in $\mathcal{NP}$ liegt.

a) Orakel rät mögliche Lösung $\mathcal{L}$ für gegebene Instanz

b) Gebe in polynomieller Zeit berechenbares Zertifikat an, dass $\mathcal{L}$ eine Lösung von I ist.

→ **Beispiel:** Algorithmus, der in polynomieller Zeit überprüft, ob geratene Lösung $\mathcal{L}$ tatsächlich eine Lösung von I ist.

2. Schritt: Zeige, dass es $\mathcal{NP}$-vollständiges Problem Π' gibt mit $\Pi' \propto \Pi$

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L heißt **$\mathcal{NP}$–vollständig**, falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

Aussage aus Vorlesung:

Falls $L_1, L_2 \in$ NP, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Schema für Beweis, dass Problem Π $\mathcal{NP}$-vollständig ist.

1. Schritt: Zeige, dass Π in $\mathcal{NP}$ liegt.

2. Schritt: Zeige, dass es $\mathcal{NP}$-vollständiges Problem Π' gibt mit $\Pi' \propto \Pi$

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L heißt **$\mathcal{NP}$–vollständig**, falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

Aussage aus Vorlesung:

Falls $L_1, L_2 \in$ NP, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Schema für Beweis, dass Problem Π $\mathcal{NP}$-vollständig ist.

1. Schritt: Zeige, dass Π in $\mathcal{NP}$ liegt.

2. Schritt: Zeige, dass es $\mathcal{NP}$-vollständiges Problem Π' gibt mit $\Pi' \propto \Pi$

a) Gebe polynomielle Reduktion $\propto$ an, so dass $\Pi' \propto \Pi$.
Zeige, dass $\propto$ tatsächlich polynomiell ist: z.B. beweise Laufzeit im O-Kalkül.

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L heißt **$\mathcal{NP}$–vollständig**, falls gilt:

- $L \in \mathcal{NP}$ und
- für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$.

Aussage aus Vorlesung:

Falls $L_1, L_2 \in$ NP, $L_1 \propto L_2$ und L_1 $\mathcal{NP}$-vollständig, dann ist auch L_2 $\mathcal{NP}$-vollständig.

Schema für Beweis, dass Problem Π $\mathcal{NP}$-vollständig ist.

1. Schritt: Zeige, dass Π in $\mathcal{NP}$ liegt.

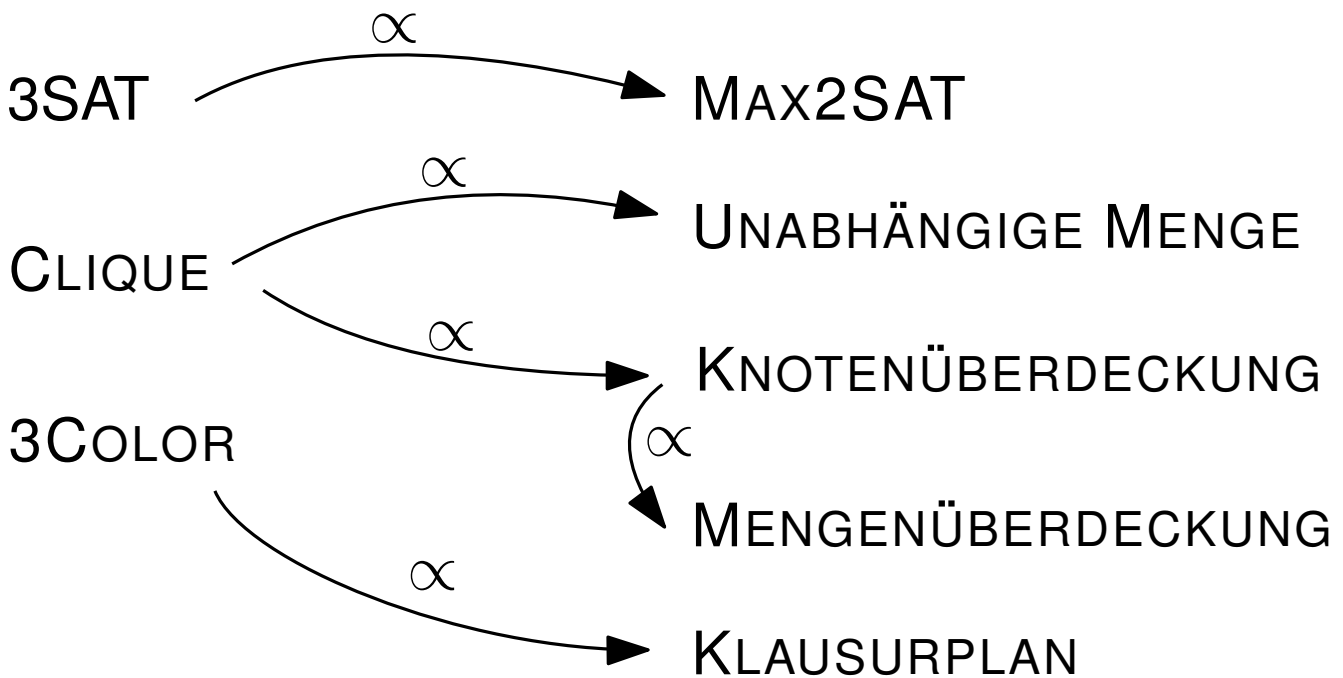
2. Schritt: Zeige, dass es $\mathcal{NP}$-vollständiges Problem Π' gibt mit $\Pi' \propto \Pi$

a) Gebe polynomielle Reduktion $\propto$ an, so dass $\Pi' \propto \Pi$.
Zeige, dass $\propto$ tatsächlich polynomiell ist: z.B. beweise Laufzeit im O-Kalkül.

b) Sei $I' \in \Pi'$ beliebige Instanz und sei $I \in \Pi$ die Transformation von I' bzgl. $\propto$.
Zeige: I' ist eine *Ja*-Instanz von Π' **genau dann wenn** I ist eine *Ja*-Instanz von Π.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Einige $\mathcal{NP}$-vollständige Probleme



MAX2SAT ist $\mathcal{NP}$-vollständig

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

1. Schritt: MAX2SAT liegt in $\mathcal{NP}$.

1. Orakel rät die Belegung der Variablen aus U.
2. Überprüfe durch Einsetzen in polynomieller Zeit, ob mindestens k Klauseln erfüllt werden.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

2. Schritt: Reduktion von 3SAT auf Max2SAT

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

i. Zeigen Sie, dass F_c nicht erfüllbar ist.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

i. Zeigen Sie, dass F_c nicht erfüllbar ist.

Wegen Gruppe I, müssen x, y, z und w_c wahr sein.

→ Klauseln aus Gruppe II können nicht wahr sein.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

ii. Geben Sie für jede Wahrheitsbelegung von x, y und z die maximale Anzahl an Klauseln von F_c an, die gleichzeitig erfüllt werden können.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

ii. Geben Sie für jede Wahrheitsbelegung von x, y und z die maximale Anzahl an Klauseln von F_c an, die gleichzeitig erfüllt werden können.

x	y	z	w_c	Anzahl erfüllter Klauseln in F_c
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

ii. Geben Sie für jede Wahrheitsbelegung von x, y und z die maximale Anzahl an Klauseln von F_c an, die gleichzeitig erfüllt werden können.

x	y	z	w_c	Anzahl erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

ii. Geben Sie für jede Wahrheitsbelegung von x, y und z die maximale Anzahl an Klauseln von F_c an, die gleichzeitig erfüllt werden können.

x	y	z	w_c	Anzahl erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Sei f eine Abbildung, die eine Klausel $c = (x \vee y \vee z)$ auf die Klauselmenge

$$F_c = \{\underbrace{x, y, z, w_c}_{\text{I}}, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_{\text{II}}, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_{\text{III}}\},$$

abbildet. Dabei ist w_c eine neu eingeführte boolesche Variable.

ii. Geben Sie für jede Wahrheitsbelegung von x, y und z die maximale Anzahl an Klauseln von F_c an, die gleichzeitig erfüllt werden können.

x	y	z	w_c	Anzahl erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

2. Schritt: Reduktion von 3SAT auf Max2SAT

- Sei I eine 3SAT-Instanz mit m Klauseln
- Bilde jede Klausel c von I auf die Klauseln F_c ab.
  ⟶ $10m$ Klauseln, die jeweils maximal 2 Literale enthalten.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

2. Schritt: Reduktion von 3SAT auf Max2SAT

- Sei I eine 3SAT-Instanz mit m Klauseln
- Bilde jede Klausel c von I auf die Klauseln F_c ab.
  ⟶ $10m$ Klauseln, die jeweils maximal 2 Literale enthalten.

Was fehlt noch für die Reduktion?

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

2. Schritt: Reduktion von 3SAT auf Max2SAT

- Sei I eine 3SAT-Instanz mit m Klauseln
- Bilde jede Klausel c von I auf die Klauseln F_c ab.
  ⟶ $10m$ Klauseln, die jeweils maximal 2 Literale enthalten.
- Setze Parameter k auf $7m$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MAX2SAT

Problem MAX2SAT: *(Skript Seite 65)*
Gegeben: Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Wahrheitbelegung, die mindestens k Klauseln erfüllt.

Zeigen Sie, dass MAX2SAT $\mathcal{NP}$-vollständig ist.

2. Schritt: Reduktion von 3SAT auf Max2SAT

- Sei I eine 3SAT-Instanz mit m Klauseln
- Bilde jede Klausel c von I auf die Klauseln F_c ab.
  ⟶ $10m$ Klauseln, die jeweils maximal 2 Literale enthalten.
- Setze Parameter k auf $7m$.
  ⟶ MAX2SAT-Instanz I' mit $10m$ Klauseln und Parameter $k = 7m$.

Beweise: I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

Max2SAT

Beweise: 3SAT-Instanz I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

x	y	z	w_c	#erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

Max2SAT

Beweise: 3SAT-Instanz I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

$\Rightarrow$ Sei I eine Ja-Instanz, d.h. jede Klausel c ist erfüllbar.

- Damit sind 7 Klauseln in F_c gleichzeitig erfüllbar.
- Folglich sind insgesamt $7m$ Klauseln von I' erfüllbar.

x	y	z	w_c	#erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Max2SAT

Beweise: 3SAT-Instanz I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

$\Rightarrow$ Sei I eine Ja-Instanz, d.h. jede Klausel c ist erfüllbar.

- Damit sind 7 Klauseln in F_c gleichzeitig erfüllbar.
- Folglich sind insgesamt $7m$ Klauseln von I' erfüllbar.

x	y	z	w_c	#erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

$\Leftarrow$ Sei I' eine *Ja*-Instanz. Es sind also mindestens $7m$ Klauseln erfüllt.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Max2SAT

Beweise: 3SAT-Instanz I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

$\Rightarrow$ Sei I eine Ja-Instanz, d.h. jede Klausel c ist erfüllbar.

- Damit sind 7 Klauseln in F_c gleichzeitig erfüllbar.
- Folglich sind insgesamt $7m$ Klauseln von I' erfüllbar.

x	y	z	w_c	#erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

$\Leftarrow$ Sei I' eine *Ja*-Instanz. Es sind also mindestens $7m$ Klauseln erfüllt.

Pro Menge F_c sind maximal 7 Klauseln gleichzeitig erfüllt

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Max2SAT

Beweise: 3SAT-Instanz I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

$\Rightarrow$ Sei I eine Ja-Instanz, d.h. jede Klausel c ist erfüllbar.

- Damit sind 7 Klauseln in F_c gleichzeitig erfüllbar.
- Folglich sind insgesamt $7m$ Klauseln von I' erfüllbar.

x	y	z	w_c	#erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

$\Leftarrow$ Sei I' eine *Ja*-Instanz. Es sind also mindestens $7m$ Klauseln erfüllt.

Pro Menge F_c sind maximal 7 Klauseln gleichzeitig erfüllt

$7m$ Klauseln von I' sind erfüllt.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Max2SAT

Beweise: 3SAT-Instanz I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

x	y	z	w_c	#erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

$\Rightarrow$ Sei I eine Ja-Instanz, d.h. jede Klausel c ist erfüllbar.

- Damit sind 7 Klauseln in F_c gleichzeitig erfüllbar.
- Folglich sind insgesamt $7m$ Klauseln von I' erfüllbar.

$\Leftarrow$ Sei I' eine *Ja*-Instanz. Es sind also mindestens $7m$ Klauseln erfüllt.

Pro Menge F_c sind maximal 7 Klauseln gleichzeitig erfüllt

$7m$ Klauseln von I' sind erfüllt.

$\Rightarrow$ In jeder Menge F_c sind 7 Klauseln erfüllt

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Max2SAT

Beweise: 3SAT-Instanz I ist erfüllbar genau dann wenn es Wahrheitsbelegung für I' gibt, die $7k$ Klauseln erfüllt.

x	y	z	w_c	#erfüllter Klauseln in F_c
0	0	0	0	6
0	0	1	0	7
0	1	0	0	7
0	1	1	0	7
1	0	0	0	7
1	0	1	0	7
1	1	0	0	7
1	1	1	1	7

$\Rightarrow$ Sei I eine Ja-Instanz, d.h. jede Klausel c ist erfüllbar.

- Damit sind 7 Klauseln in F_c gleichzeitig erfüllbar.
- Folglich sind insgesamt $7m$ Klauseln von I' erfüllbar.

$\Leftarrow$ Sei I' eine *Ja*-Instanz. Es sind also mindestens $7m$ Klauseln erfüllt.

Pro Menge F_c sind maximal 7 Klauseln gleichzeitig erfüllt

$7m$ Klauseln von I' sind erfüllt.

$\Rightarrow$ In jeder Menge F_c sind 7 Klauseln erfüllt

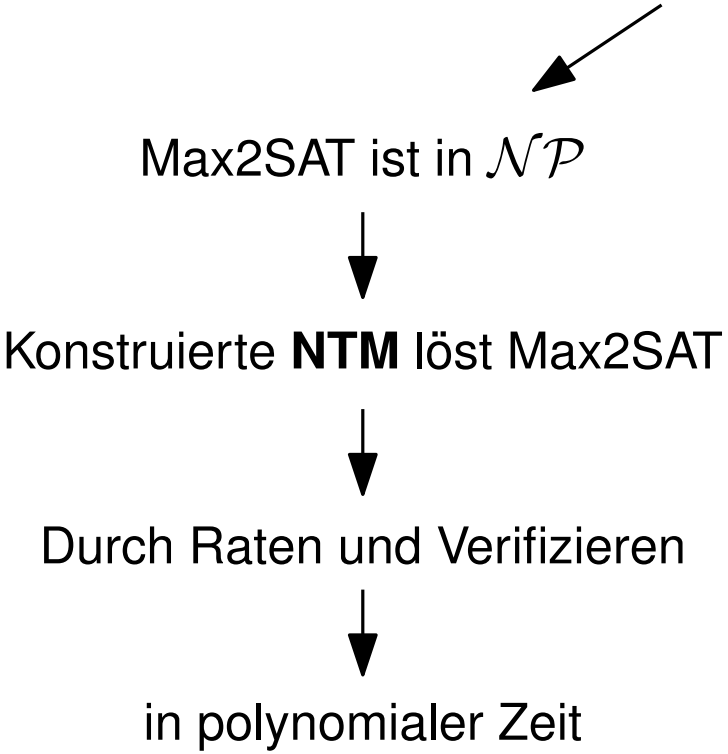
aus folgt das jede Klausel c von I ebenfalls erfüllt.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Max2SAT

Behauptung: Max2SAT ist $\mathcal{NP}$-vollständig.

Zusammenfassung des Beweises



Max2SAT

Behauptung: Max2SAT ist $\mathcal{NP}$-vollständig.

Zusammenfassung des Beweises

Max2SAT ist in $\mathcal{NP}$

Konstruierte **NTM** löst Max2SAT

Durch Raten und Verifizieren

in polynomialer Zeit

Max2SAT ist $\mathcal{NP}$-schwer

Polynomielle Reduktion von 3SAT

aus 3SAT-Instanz I konstruiere Max2SAT-Instanz I'

Zeit für Konstruktion von I' ist poly($|I|$)

I' lösbar gdw. I lösbar

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Komplexitätsklassen und Werkzeugkasten

Die Klasse $\mathcal{NP}$:
= alle Probleme, die von einer **NTM** in polynomialer Zeit gelöst werden

Die Klasse $\mathcal{P}$:
= alle Probleme, die von einer **DTM** in polynomialer Zeit gelöst werden

$\mathcal{NP}$-schwere Probleme:
= alle Probleme Π, so dass für alle Probleme in $\Pi' \in \mathcal{NP}$ Π' polynomial reduzierbar auf Π d.h. $\Pi' \propto \Pi$.

Ist $\Pi \in \mathcal{P}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{P}$.

(2) Konstruiere **DTM** die Π löst.

Ist $\Pi \in \mathcal{NP}$?

(1) Zeige $\Pi \propto \Pi'$ für bekanntes $\Pi' \in \mathcal{NP}$.

(2) Konstruiere **NTM** die Π löst.

Ist Π $\mathcal{NP}$-schwer?

(1) Zeige $\Pi' \propto \Pi$ für bekanntes Π' $\mathcal{NP}$-schwer.

(2) Beweise ad hoc, dass Π $\mathcal{NP}$-schwer.

gute Idee

schlechte Idee

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Tipps für Reduktion auf Max2Color

Problem MAX2COLOR:
Gegeben: Graph G, Zahl k
Frage: Existiert eine 2-Färbung der Knoten von G, so dass mindestens k Kanten von G Endpunkte in beiden Farben haben?

Behauptung auf Übungsblatt 4: Max2Color ist $\mathcal{NP}$-vollständig.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Tipps für Reduktion auf Max2Color

Problem MAX2COLOR:
Gegeben: Graph G, Zahl k
Frage: Existiert eine 2-Färbung der Knoten von G, so dass mindestens k Kanten von G Endpunkte in beiden Farben haben?

Behauptung auf Übungsblatt 4: Max2Color ist $\mathcal{NP}$-vollständig.

Für eine polynomielle Reduktion $\Pi \propto$ Max2Color:

- Starte mit Instanz I des Problems Π
- Konstruiere Instanz I' von Max2Color (Graphen G, Zahl k)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Tipps für Reduktion auf Max2Color

Problem MAX2COLOR:
Gegeben: Graph G, Zahl k
Frage: Existiert eine 2-Färbung der Knoten von G, so dass mindestens k Kanten von G Endpunkte in beiden Farben haben?

Behauptung auf Übungsblatt 4: Max2Color ist $\mathcal{NP}$-vollständig.

Für eine polynomielle Reduktion $\Pi \propto$ Max2Color:

- Starte mit Instanz I des Problems Π
- Konstruiere Instanz I' von Max2Color (Graphen G, Zahl k)

Tipp:

zwei Knoten die unbedingt verschiedene Farben erhalten sollen

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Tipps für Reduktion auf Max2Color

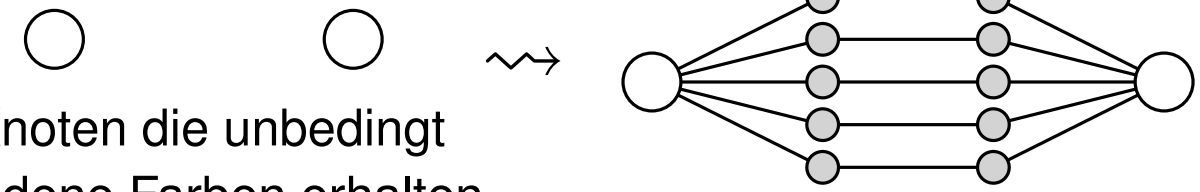
Problem MAX2COLOR:
Gegeben: Graph G, Zahl k
Frage: Existiert eine 2-Färbung der Knoten von G, so dass mindestens k Kanten von G Endpunkte in beiden Farben haben?

Behauptung auf Übungsblatt 4: Max2Color ist $\mathcal{NP}$-vollständig.

Für eine polynomielle Reduktion $\Pi \propto$ Max2Color:

- Starte mit Instanz I des Problems Π
- Konstruiere Instanz I' von Max2Color (Graphen G, Zahl k)

Tipp:



zwei Knoten die unbedingt verschiedene Farben erhalten sollen

w Übergänge

gleiche Farbe:
$\leq 2w$ gute Kanten

verschiedene Farben:
$\geq 3w$ gute Kanten

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Tipps für Reduktion auf Max2Color

Problem MAX2COLOR:
Gegeben: Graph G, Zahl k
Frage: Existiert eine 2-Färbung der Knoten von G, so dass mindestens k Kanten von G Endpunkte in beiden Farben haben?

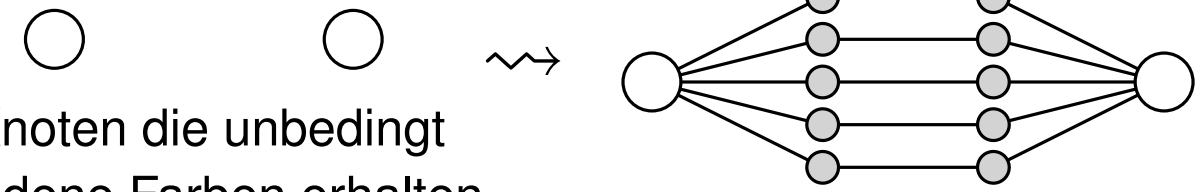
Behauptung auf Übungsblatt 4: Max2Color ist $\mathcal{NP}$-vollständig.

Für eine polynomielle Reduktion $\Pi \propto$ Max2Color:

- Starte mit Instanz I des Problems Π
- Konstruiere Instanz I' von Max2Color (Graphen G, Zahl k)

Wähle k, so dass $2w$ gute Kanten zu wenig ist.

Tipp:



zwei Knoten die unbedingt verschiedene Farben erhalten sollen

w Übergänge

gleiche Farbe:
$\leq 2w$ gute Kanten

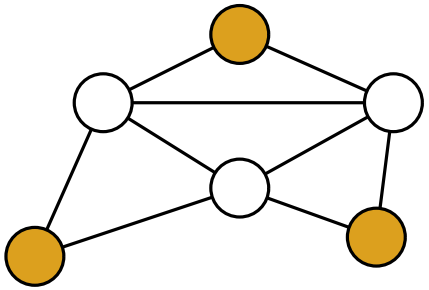
verschiedene Farben:
$\geq 3w$ gute Kanten

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

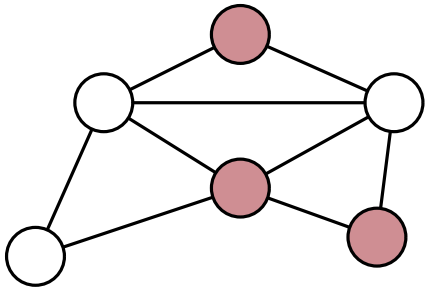
UNABHÄNGIGE MENGE ist $\mathcal{NP}$-vollständig

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge

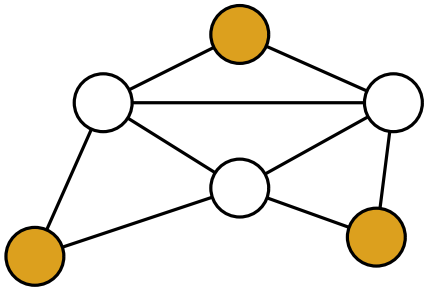


Keine unabhängige Menge

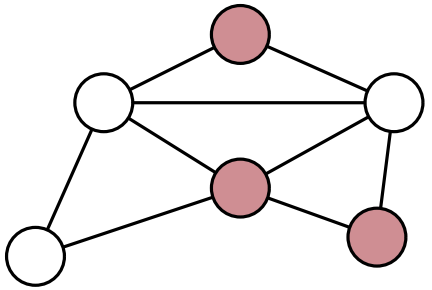
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge

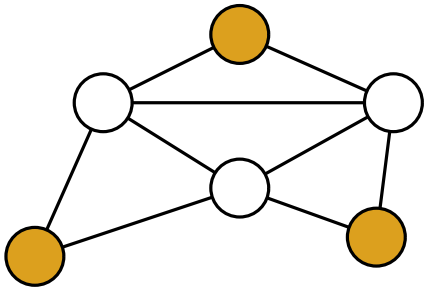


Keine unabhängige Menge

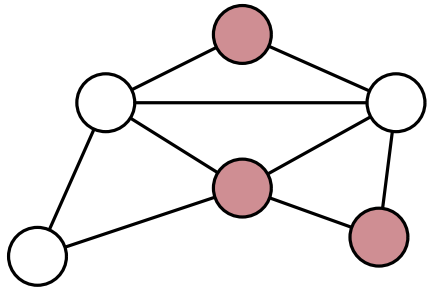
1. Schritt: UNABHÄNGIGE MENGE liegt in $\mathcal{NP}$.

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge



Keine unabhängige Menge

1. Schritt: UNABHÄNGIGE MENGE liegt in $\mathcal{NP}$.

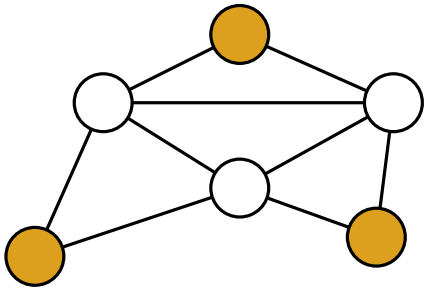
- Orakel rät Knotenmenge V'.
- Überprüfe, ob V' unabhängige Menge von G ist mit $k \leq |V'|$.
  - $|V'| \geq k$
  - V' sind Knoten aus V
  - Es gibt keine zwei Knoten $u, v \in V'$ mit $\{u, v\} \in E$.

Laufzeit $O(n^2)$

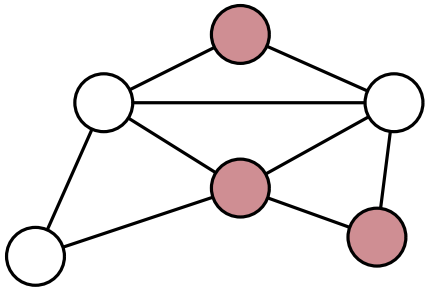
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge

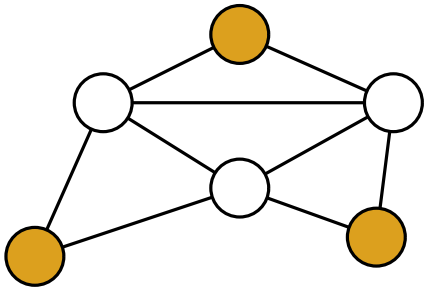


Keine unabhängige Menge

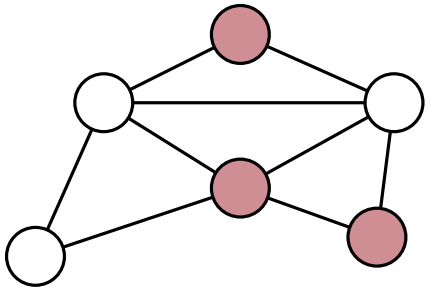
2. Schritt: UNABHÄNGIGE MENGE ist $\mathcal{NP}$-schwer.

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge



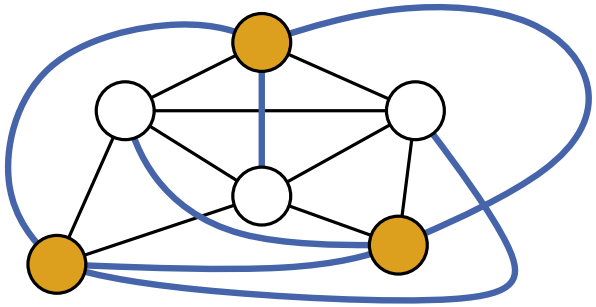
Keine unabhängige Menge

2. Schritt: UNABHÄNGIGE MENGE ist $\mathcal{NP}$-schwer.

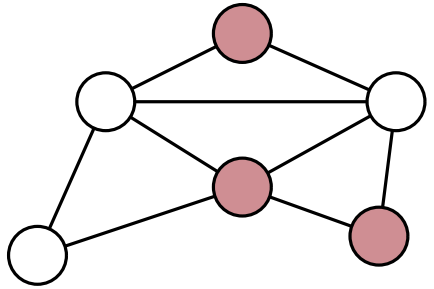
Problem CLIQUE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine Clique C in G mit $|C| \geq k$?
Hinweis: $C \subseteq V$ heißt *Clique*, falls für jedes Paar $u, v \in C$ die Kante $\{u, v\} \in E$ existiert.

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge



Keine unabhängige Menge

2. Schritt: UNABHÄNGIGE MENGE ist $\mathcal{NP}$-schwer. CLIQUE $\propto$ UNABHÄNGIGE MENGE

Sei $I = (G = (V, E), k)$ Instanz von Clique.

Erstelle Komplementgraph $\overline{G} = (V, \overline{E})$ mit $\{u, v\} \in \overline{E} \Leftrightarrow \{u, v\} \notin E$

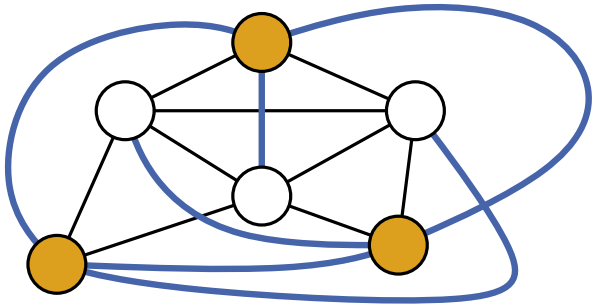
Instanz für UNABHÄNGIGE MENGE ist $I' = (\overline{G}, k)$

Kopieren des Graphen + $\overline{E}$ erstellen $\rightarrow$ Laufzeit $O(|V|^2)$

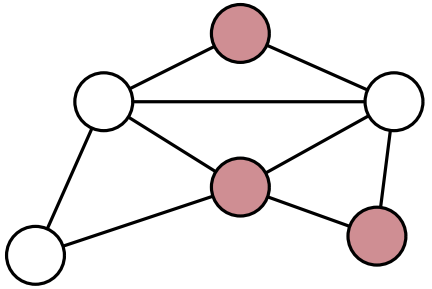
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge



Keine unabhängige Menge

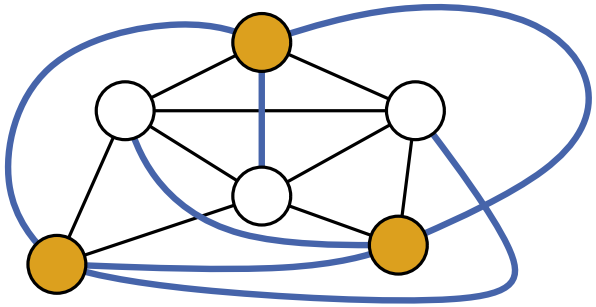
2. Schritt: UNABHÄNGIGE MENGE ist $\mathcal{NP}$-schwer.

Reduktion ist korrekt: G hat Clique C mit $|C| \geq k$. $\Leftrightarrow$
$\overline{G}$ hat unabhängige Menge V' mit $|V'| \geq k$

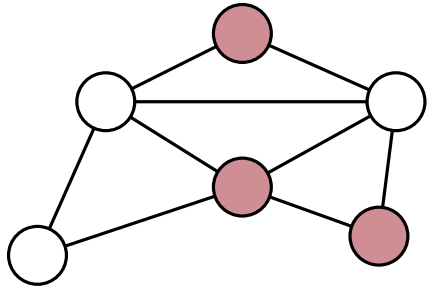
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Unabhängige Menge

Problem UNABHÄNGIGE MENGE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Existiert eine unabhängige Knotenmenge $V' \subseteq V$, so dass $|V'| \geq k$ gilt?
Hinweis: $V' \subseteq V$ heißt *unabhängig*, falls für alle $u, v \in V'$ mit $u \neq v$ gilt $\{u, v\} \notin E$.



Unabhängige Menge



Keine unabhängige Menge

2. Schritt: UNABHÄNGIGE MENGE ist $\mathcal{NP}$-schwer.

Reduktion ist korrekt: G hat Clique C mit $|C| \geq k$. $\Leftrightarrow$
$\overline{G}$ hat unabhängige Menge V' mit $|V'| \geq k$

"$\Rightarrow$" Knoten aus C sind in $\overline{G}$ nicht miteinander verbunden $\rightarrow$ C ist unabhängige Menge in $\overline{G}$ mit $|C| \geq k$

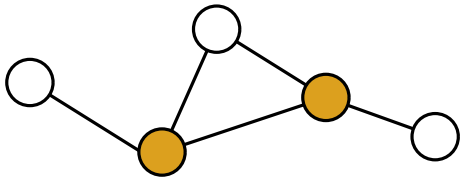
"$\Leftarrow$" Knoten aus V' sind in G miteinander verbunden $\rightarrow$ V' ist Clique in G mit $V' \geq k$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

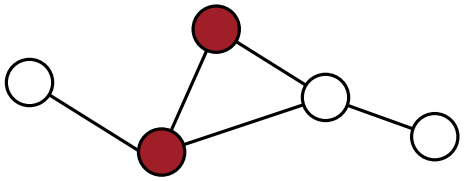
KNOTENÜBERDECKUNG ist $\mathcal{NP}$-vollständig

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung

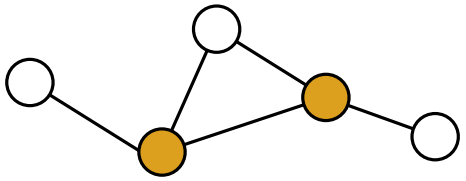


Keine Knotenüberdeckung

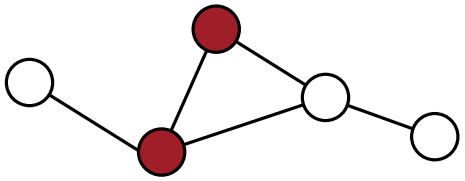
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung



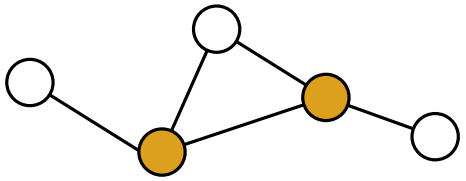
Keine Knotenüberdeckung

1. Schritt: KNOTENÜBERDECKUNG liegt in $\mathcal{NP}$

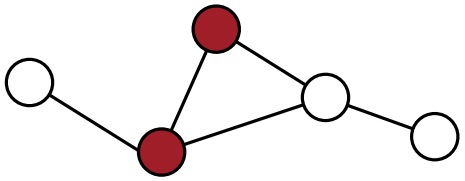
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung



Keine Knotenüberdeckung

1. Schritt: KNOTENÜBERDECKUNG liegt in $\mathcal{NP}$

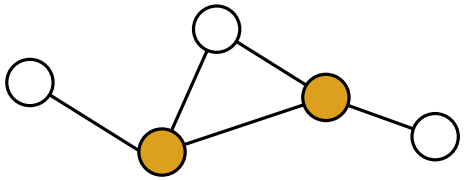
- Orakel rät Knotenmenge V'.
- Überprüfe, ob V' Knotenüberdeckung von G ist mit $|V'| \leq k$.
  - $|V'| \leq k$
  - V' sind Knoten aus V
  - Für jede Kante $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.

Laufzeit $O(n^3)$

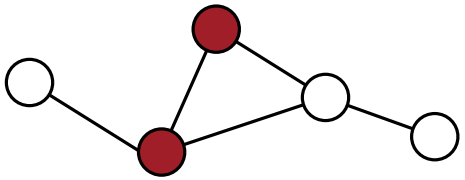
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung

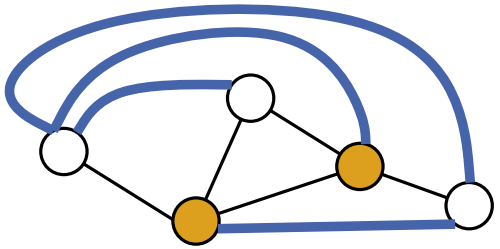


Keine Knotenüberdeckung

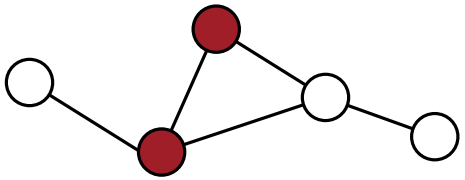
2. Schritt: KNOTENÜBERDECKUNG ist $\mathcal{NP}$-schwer. CLIQUE $\propto$ KNOTENÜBERDECKUNG

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung



Keine Knotenüberdeckung

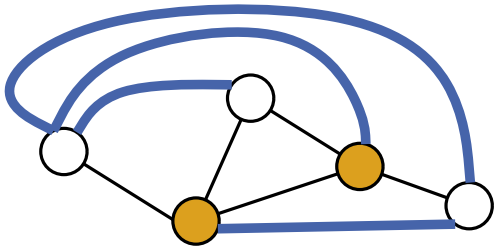
2. Schritt: KNOTENÜBERDECKUNG ist $\mathcal{NP}$-schwer. CLIQUE $\propto$ KNOTENÜBERDECKUNG

Lemma: Sei $V' \subseteq V$. $C = V \setminus V'$ ist Clique in $G \Leftrightarrow V'$ ist Knotenüberdeckung von $\overline{G}$

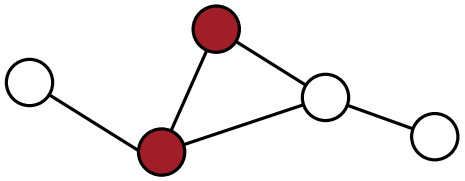
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung



Keine Knotenüberdeckung

2. Schritt: KNOTENÜBERDECKUNG ist $\mathcal{NP}$-schwer. CLIQUE $\propto$ KNOTENÜBERDECKUNG

Lemma: Sei $V' \subseteq V$. $C = V \setminus V'$ ist Clique in $G \Leftrightarrow V'$ ist Knotenüberdeckung von $\overline{G}$

Beweis:

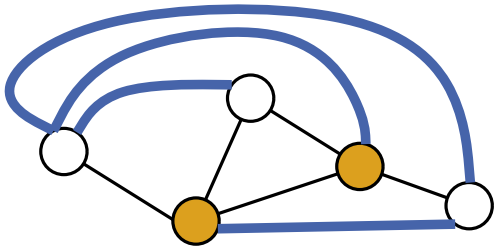
"$\Rightarrow$" Angenommen es gibt $\{u, v\} \in \overline{E}$, mit $u \notin V'$ und $v' \notin V'$.

Es gilt also $u \in C$ und $v \in C$. $\Rightarrow$ $\{u, v\} \in E$, weil C Clique von G  Def. $\overline{G}$

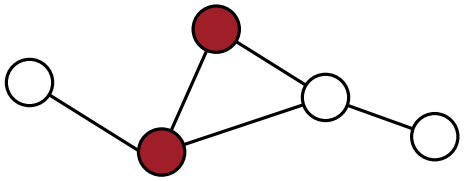
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung



Keine Knotenüberdeckung

2. Schritt: KNOTENÜBERDECKUNG ist $\mathcal{NP}$-schwer. CLIQUE $\propto$ KNOTENÜBERDECKUNG

Lemma: Sei $V' \subseteq V$. $C = V \setminus V'$ ist Clique in $G \Leftrightarrow V'$ ist Knotenüberdeckung von $\overline{G}$

Beweis:

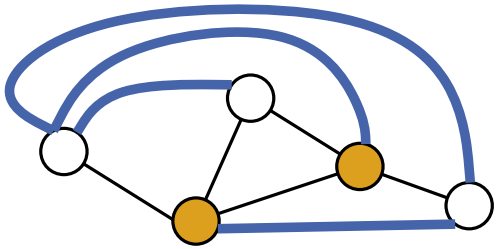
"$\Rightarrow$" Angenommen es gibt $\{u, v\} \in \overline{E}$, mit $u \notin V'$ und $v' \notin V'$.
Es gilt also $u \in C$ und $v \in C$. $\Rightarrow$ $\{u, v\} \in E$, weil C Clique von G  Def. $\overline{G}$

"$\Leftarrow$" Seien $u, v \in V \setminus V'$. Es gibt keine Kante $\{u, v\} \in \overline{E}$, sonst ist V' keine Knotenüb.
Damit gibt es die Kante $\{u, v\}$ in G.

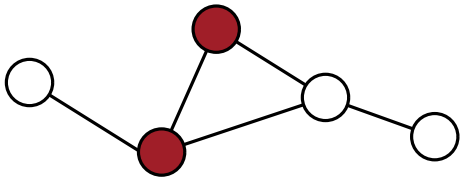
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung



Keine Knotenüberdeckung

2. Schritt: KNOTENÜBERDECKUNG ist $\mathcal{NP}$-schwer. CLIQUE $\propto$ KNOTENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von CLIQUE.

Erstelle Komplementgraph $\overline{G} = (V, \overline{E})$ mit $\{u, v\} \in \overline{E} \Leftrightarrow \{u, v\} \notin E$

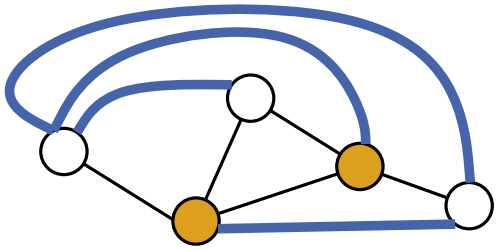
Instanz für KNOTENÜBERDECKUNG ist $I' = (\overline{G}, |V| - k)$

Kopieren des Graphen + $\overline{E}$ erstellen $\rightarrow$ Laufzeit $O(|V|^2)$

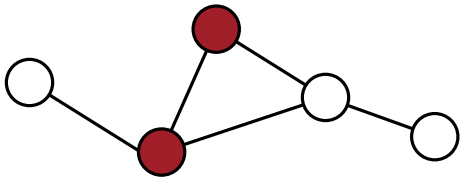
Lemma: Sei $V' \subseteq V$. $C = V \setminus V'$ ist Clique in $G \Leftrightarrow V'$ ist Knotenüberdeckung von $\overline{G}$

Knotenüberdeckung

Problem KNOTENÜBERDECKUNG
Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenüberdeckung $V' \subseteq V$ mit $|V'| \leq k$?
Hinweis: $V' \subseteq V$ heißt *Knotenüberdeck.*, falls für alle $\{u, v\} \in E$ gilt $u \in V'$ oder $v \in V'$.



Knotenüberdeckung



Keine Knotenüberdeckung

2. Schritt: KNOTENÜBERDECKUNG ist $\mathcal{NP}$-schwer. CLIQUE $\propto$ KNOTENÜBERDECKUNG

Reduktion ist korrekt: G hat Clique C mit $|C| \geq k$. $\Leftrightarrow$
$\overline{G}$ hat Knotenüberdeckung V' mit $|V'| \leq |V| - k$

Korrektheit folgt direkt aus Lemma.

Lemma: Sei $V' \subseteq V$. $C = V \setminus V'$ ist Clique in $G \Leftrightarrow V'$ ist Knotenüberdeckung von $\overline{G}$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MENGENÜBERDECKUNG ist $\mathcal{NP}$-vollständig

Mengenüberdeckung

Problem MENGENÜBERDECKUNG
Gegeben: Universum $\mathcal{U} = \{x_1, \ldots, x_m\}$, Teilmengen $S_1, \ldots, S_n \subseteq \mathcal{U}$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$, sodass $\bigcup_{i \in C} S_i = \mathcal{U}$?
Hinweis: Verwenden Sie das Problem KNOTENÜBERDECKUNG für die Reduktion.

Beispiel: $\mathcal{U} = \{1, 2, 3, 4, 5\}$

$S_1 = \{1, 2, 3\}$ $S_2 = \{2, 4\}$ $S_3 = \{3, 4\}$ $S_4 = \{4, 5\}$

k=4: $\mathcal{U} = S_1 \cup S_2 \cup S_3 \cup S_4$

k=2: $\mathcal{U} = S_1 \cup S_4$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Problem MENGENÜBERDECKUNG
Gegeben: Universum $\mathcal{U} = \{x_1, \ldots, x_m\}$, Teilmengen $S_1, \ldots, S_n \subseteq \mathcal{U}$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$, sodass $\bigcup_{i \in C} S_i = \mathcal{U}$?
Hinweis: Verwenden Sie das Problem KNOTENÜBERDECKUNG für die Reduktion.

1. Schritt: MENGENÜBERDECKUNG liegt in $\mathcal{NP}$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Problem MENGENÜBERDECKUNG
Gegeben: Universum $\mathcal{U} = \{x_1, \ldots, x_m\}$, Teilmengen $S_1, \ldots, S_n \subseteq \mathcal{U}$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$, sodass $\bigcup_{i \in C} S_i = \mathcal{U}$?
Hinweis: Verwenden Sie das Problem KNOTENÜBERDECKUNG für die Reduktion.

1. Schritt: MENGENÜBERDECKUNG liegt in $\mathcal{NP}$.

- Orakel rät Zahlen $i_1, \ldots, i_\ell$.
- Überprüfe
  - Nicht zu viele Zahlen: $\ell \leq k$
  - Zahlen im richtigen Zahlenbereich: $1 \leq i_j \leq n$ für alle $1 \leq j \leq \ell$
  - $\bigcup_{j=1}^{\ell} S_{i_j} = \mathcal{U}$

In polynomieller Zeit berechenbar.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Problem MENGENÜBERDECKUNG
Gegeben: Universum $\mathcal{U} = \{x_1, \ldots, x_m\}$, Teilmengen $S_1, \ldots, S_n \subseteq \mathcal{U}$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$, sodass $\bigcup_{i \in C} S_i = \mathcal{U}$?
Hinweis: Verwenden Sie das Problem KNOTENÜBERDECKUNG für die Reduktion.

2. Schritt: MENGENÜBERDECKUNG ist $\mathcal{NP}$-schwer.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Problem MENGENÜBERDECKUNG
Gegeben: Universum $\mathcal{U} = \{x_1, \ldots, x_m\}$, Teilmengen $S_1, \ldots, S_n \subseteq \mathcal{U}$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$, sodass $\bigcup_{i \in C} S_i = \mathcal{U}$?
Hinweis: Verwenden Sie das Problem KNOTENÜBERDECKUNG für die Reduktion.

2. Schritt: MENGENÜBERDECKUNG ist $\mathcal{NP}$-schwer.

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Problem MENGENÜBERDECKUNG
Gegeben: Universum $\mathcal{U} = \{x_1, \ldots, x_m\}$, Teilmengen $S_1, \ldots, S_n \subseteq \mathcal{U}$ und Zahl $k \in \mathbb{N}$.
Frage: Gibt es eine Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$, sodass $\bigcup_{i \in C} S_i = \mathcal{U}$?
Hinweis: Verwenden Sie das Problem KNOTENÜBERDECKUNG für die Reduktion.

2. Schritt: MENGENÜBERDECKUNG ist $\mathcal{NP}$-schwer.

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Rightarrow$" G besitzt Knotenüberdeckung V' der Größe $\ell = |V| \leq k$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Rightarrow$" G besitzt Knotenüberdeckung V' der Größe $\ell = |V| \leq k$

Knoten induzieren Menge C an Indizes. **Zeige:** $\bigcup_{i \in C} S_i = \mathcal{U}$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Rightarrow$" G besitzt Knotenüberdeckung V' der Größe $\ell = |V| \leq k$

Knoten induzieren Menge C an Indizes. **Zeige:** $\bigcup_{i \in C} S_i = \mathcal{U}$

Betrachte Kante $e \in E = \mathcal{U}$. Es gibt Knoten v in V', so dass e inzident zu v ist.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Rightarrow$" G besitzt Knotenüberdeckung V' der Größe $\ell = |V| \leq k$

Knoten induzieren Menge C an Indizes. **Zeige:** $\bigcup_{i \in C} S_i = \mathcal{U}$

Betrachte Kante $e \in E = \mathcal{U}$. Es gibt Knoten v in V', so dass e inzident zu v ist.

Sei i der Index von v, somit enthält S_i die Kante e.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Leftarrow$" Es gibt Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$ und $\bigcup_{i \in C} S_i = \mathcal{U}$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Leftarrow$" Es gibt Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$ und $\bigcup_{i \in C} S_i = \mathcal{U}$.

Jede Menge S_i mit $i \in C$ korrespondiert mit einem Knoten v_i.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Leftarrow$" Es gibt Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$ und $\bigcup_{i \in C} S_i = \mathcal{U}$.

Jede Menge S_i mit $i \in C$ korrespondiert mit einem Knoten v_i.

Sei V' die Menge der Knoten induziert duch C, es gilt $|V'| \leq m = k$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Leftarrow$" Es gibt Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$ und $\bigcup_{i \in C} S_i = \mathcal{U}$.

Jede Menge S_i mit $i \in C$ korrespondiert mit einem Knoten v_i.

Sei V' die Menge der Knoten induziert duch C, es gilt $|V'| \leq m = k$.

Betrachte beliebige Kante $e \in \mathcal{U}$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Leftarrow$" Es gibt Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$ und $\bigcup_{i \in C} S_i = \mathcal{U}$.

Jede Menge S_i mit $i \in C$ korrespondiert mit einem Knoten v_i.

Sei V' die Menge der Knoten induziert duch C, es gilt $|V'| \leq m = k$.

Betrachte beliebige Kante $e \in \mathcal{U}$.

Da C Mengenüberdeckung von $\mathcal{U}$ ist, gibt es $i \in C$ mit $e \in S_i$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mengenüberdeckung

Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

Sei $I = (G = (V, E), k)$ Instanz von KNOTENÜBERDECKUNG, mit $V = \{v_1, \ldots, v_n\}$.

1. Sei $\mathcal{U} = E$
2. Def. n Teilmengen $S_1, \ldots, S_n$ von $\mathcal{U}$: S_i enthält alle Kanten, die zu v_i inzident sind.
3. Setze $k = m$

"$\Leftarrow$" Es gibt Menge $C \subseteq \{1, \ldots, n\}$ mit $|C| \leq k$ und $\bigcup_{i \in C} S_i = \mathcal{U}$.

Jede Menge S_i mit $i \in C$ korrespondiert mit einem Knoten v_i.

Sei V' die Menge der Knoten induziert duch C, es gilt $|V'| \leq m = k$.

Betrachte beliebige Kante $e \in \mathcal{U}$.

Da C Mengenüberdeckung von $\mathcal{U}$ ist, gibt es $i \in C$ mit $e \in S_i$.

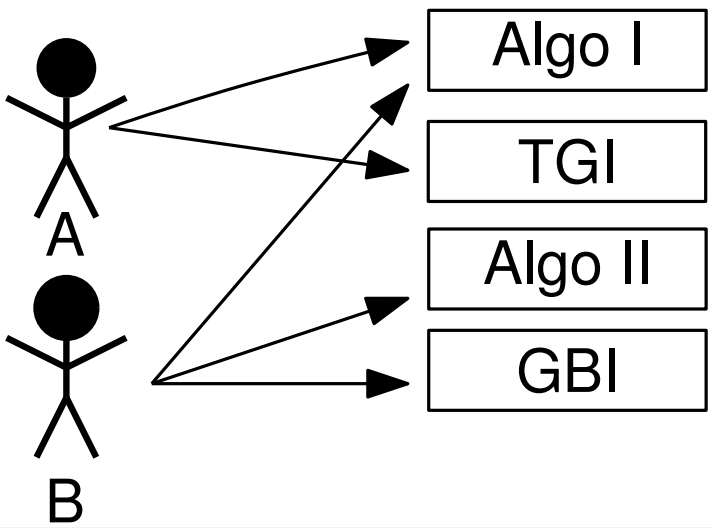
Nach Definition ist v_i inzident zu e und $v_i \in V' \Rightarrow V'$ ist Knotenüberdeckung

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

KLAUSURPLAN ist $\mathcal{NP}$-vollständig

Klausurplan

Beispiel:



	Montag	Diens
1. Block		
2. Block		
3. Block		
4. Block		
. . .		

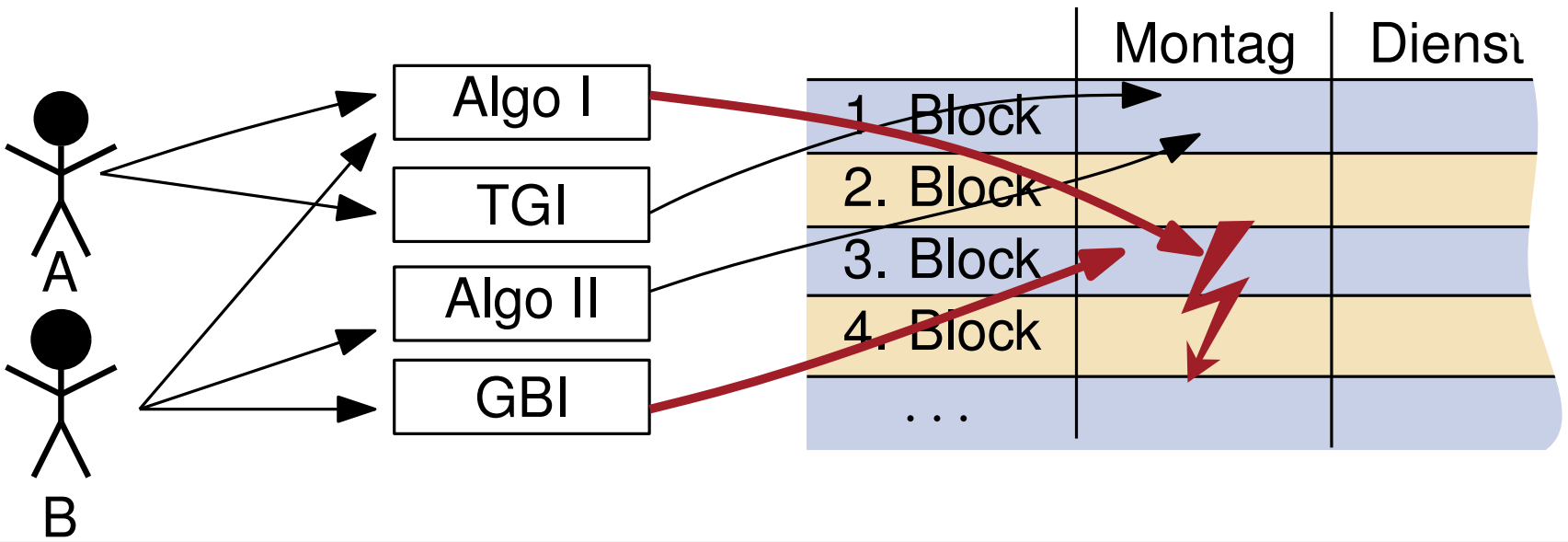
Klausurplan

Problem KLAUSURPLAN

Gegeben:
- Menge K an Klausuren
- Menge S an Studenten, die an Klausuren teilnehmen.
- Mögliche Zeitbereiche
- Parameter k.

Frage: Gibt es Zuweisung der Klausuren auf Zeitbereiche, sodass es maximal k Konflikte gibt?

Beispiel:



Klausurplan

Problem KLAUSURPLAN

Gegeben:
- Menge K an Klausuren
- Menge S an Studenten, die an Klausuren teilnehmen.
- Mögliche Zeitbereiche
- Parameter k.

Frage: Gibt es Zuweisung der Klausuren auf Zeitbereiche, sodass es maximal k Konflikte gibt?

Klausurplan

Problem KLAUSURPLAN

Gegeben:
- Menge K an Klausuren
- Menge S an Studenten, die an Klausuren teilnehmen.
- Mögliche Zeitbereiche
- Parameter k.

Frage: Gibt es Zuweisung der Klausuren auf Zeitbereiche, sodass es maximal k Konflikte gibt?

1. Schritt: Zeige, dass Klausurplan in $\mathcal{NP}$ liegt.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Problem KLAUSURPLAN

Gegeben:
- Menge K an Klausuren
- Menge S an Studenten, die an Klausuren teilnehmen.
- Mögliche Zeitbereiche
- Parameter k.

Frage: Gibt es Zuweisung der Klausuren auf Zeitbereiche, sodass es maximal k Konflikte gibt?

1. Schritt: Zeige, dass Klausurplan in $\mathcal{NP}$ liegt.

- Orakel rät Zuweisung der Klausuren.
- Überprüfe
  - Zuweisung ist gültig.
  - Berechne Konflikte $O(|S||K^2|)$ Zeit
  - Überprüfe, dass Anzahl Konflikte $\leq k$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Problem KLAUSURPLAN

Gegeben:
- Menge K an Klausuren
- Menge S an Studenten, die an Klausuren teilnehmen.
- Mögliche Zeitbereiche
- Parameter k.

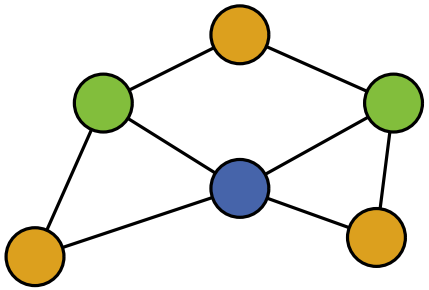
Frage: Gibt es Zuweisung der Klausuren auf Zeitbereiche, sodass es maximal k Konflikte gibt?

2. Schritt: Zeige, dass KLAUSURPLAN NP-schwer ist. 3COLOR $\propto$ KLAUSURPLAN

Problem 3COLOR

Gegeben: Graph $G = (V, E)$ und Parameter k

Frage: Gibt es eine Knotenfärbung von G mit höchstens k Farben, so dass je zwei adjazente Knoten verschiedene Farben besitzen?

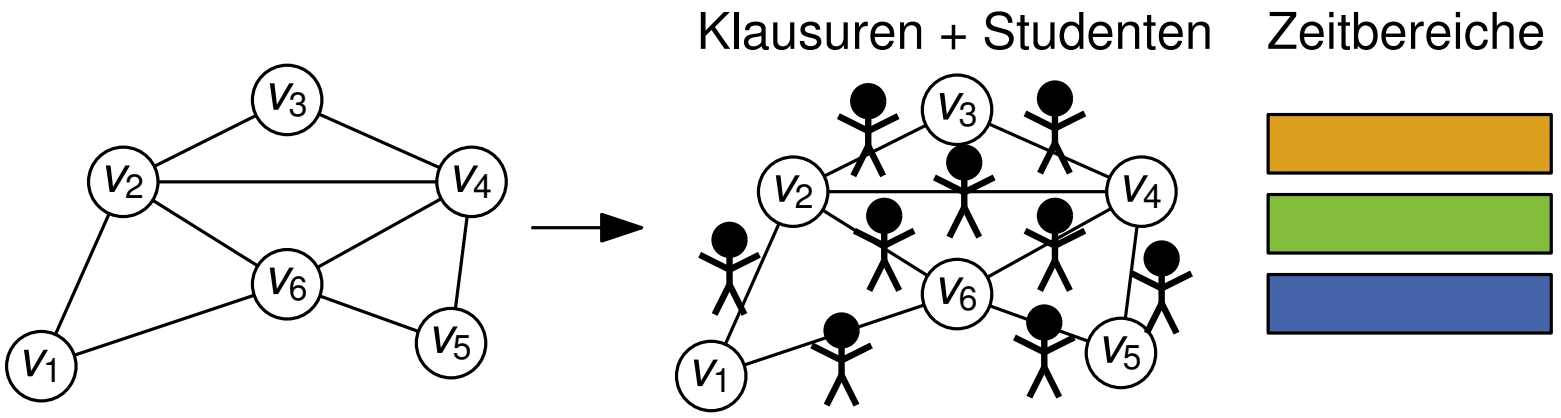


Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.



Problem 3COLOR
Gegeben: Graph $G = (V, E)$
Frage: Gibt es eine Knotenfärbung von G mit höchstens 3 Farben, so dass je zwei adjazente Knoten verschiedene Farben besitzen?

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Rightarrow$" Die Instanz $I = (G = (V, E))$ ist 3 färbbar

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Rightarrow$" Die Instanz $I = (G = (V, E))$ ist 3 färbbar

Die gegebene Färbung ordnet jeder Klausur einen Zeitbereich zu.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Rightarrow$" Die Instanz $I = (G = (V, E))$ ist 3 färbbar

Die gegebene Färbung ordnet jeder Klausur einen Zeitbereich zu.

Klausuren, die in G durch eine Kante verbunden sind, werden unterschiedlichen Zeitbereichen zugewiesen.

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Rightarrow$" Die Instanz $I = (G = (V, E))$ ist 3 färbbar

Die gegebene Färbung ordnet jeder Klausur einen Zeitbereich zu.

Klausuren, die in G durch eine Kante verbunden sind, werden unterschiedlichen Zeitbereichen zugewiesen.

Studenten entsprechen genau diesen Kanten.

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Rightarrow$" Die Instanz $I = (G = (V, E))$ ist 3 färbbar

Die gegebene Färbung ordnet jeder Klausur einen Zeitbereich zu.

Klausuren, die in G durch eine Kante verbunden sind, werden unterschiedlichen Zeitbereichen zugewiesen.

Studenten entsprechen genau diesen Kanten.

→ Klausurenplan hat keine Konflikte.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Leftarrow$" Der Konstruierte Klausurenplan hat keine Konflikte

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Leftarrow$" Der Konstruierte Klausurenplan hat keine Konflikte

Jeder Zeitbereich entspricht einer Farbe.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Leftarrow$" Der Konstruierte Klausurenplan hat keine Konflikte

Jeder Zeitbereich entspricht einer Farbe.

Jede Klausur entspricht einem Knoten.

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Leftarrow$" Der Konstruierte Klausurenplan hat keine Konflikte

Jeder Zeitbereich entspricht einer Farbe.
Jede Klausur entspricht einem Knoten.

$\mapsto$ Zuordnung Klausuren zu Zeitbereichen liefert Färbung der Knoten.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Leftarrow$" Der Konstruierte Klausurenplan hat keine Konflikte

Jeder Zeitbereich entspricht einer Farbe.
Jede Klausur entspricht einem Knoten.
$\mapsto$ Zuordnung Klausuren zu Zeitbereichen liefert Färbung der Knoten.

Jeder Student nimmt nur an Klausuren teil, die nicht zur selben Zeit statt finden.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Leftarrow$" Der Konstruierte Klausurenplan hat keine Konflikte

Jeder Zeitbereich entspricht einer Farbe.
Jede Klausur entspricht einem Knoten.
$\mapsto$ Zuordnung Klausuren zu Zeitbereichen liefert Färbung der Knoten.

Jeder Student nimmt nur an Klausuren teil, die nicht zur selben Zeit statt finden.
Jeder Student entspricht einer Kante in G.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Klausurplan

Reduktion: Gegeben Instanz $I = (G = (V, E))$ von 3COLOR.

- Für jeden Knoten $u \in V$ führe Klausur u ein.
- Für jede Farbe führe einen Zeitbereich ein.
- Für jede Kante $\{u, v\} \in E$ führe Student ein, der an u und v teilnehmen möchte.

Setze $k = 0$.

"$\Leftarrow$" Der Konstruierte Klausurenplan hat keine Konflikte

Jeder Zeitbereich entspricht einer Farbe.
Jede Klausur entspricht einem Knoten.
$\mapsto$ Zuordnung Klausuren zu Zeitbereichen liefert Färbung der Knoten.

Jeder Student nimmt nur an Klausuren teil, die nicht zur selben Zeit statt finden.
Jeder Student entspricht einer Kante in G.

$\longrightarrow$ Benachbarte Knoten in G haben unterschiedliche Färbungen.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Reduktionsschemas

Beschränkung eines Problems

- offensichtliche 1 zu 1 Beziehung
- 3SAT $\propto$ 4SAT

Lokales Ersetzen

- Veränderung der lokalen Struktur
- weitesgehend unabhängige Veränderung
- 3SAT $\propto$ MAX2SAT

Komponenten Design

- Modulierung von Interaktion zwischen Komponenten
- Modeliere z.B. Literale und Klauseln
- 3SAT $\propto$ 3COLOR

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$$\mathcal{NP} = co - \mathcal{NP}?$$

$\mathcal{NP}$ = **co-$\mathcal{NP}$?**

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$\mathcal{NP}$ = **co-**$\mathcal{NP}$**?**

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Wiederholung:

Die Klasse $\mathcal{NP}$ ist die Menge aller Sprachen L, für die es eine nichtdeterministische Turingmaschine gibt, die L in polynomieller Zeit erkennt.

Die Klasse **co**-$\mathcal{NP}$ ist die Menge aller Sprachen deren Komplement in $\mathcal{NP}$ enthalten ist:

$$\text{co-}\mathcal{NP} = \{L \subseteq \Sigma^\star \mid \Sigma^\star \setminus L \in \text{NP}\}$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$\mathcal{NP}$ = **co-$\mathcal{NP}$?**

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Sei Π_1 ein $\mathcal{NP}$-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$\mathcal{NP}$ = **co-$\mathcal{NP}$?**

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Sei Π_1 ein $\mathcal{NP}$-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathcal{NP}$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$\mathcal{NP}$ = **co-$\mathcal{NP}$?**

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Sei Π_1 ein $\mathcal{NP}$-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathcal{NP}$

Zeige: Komplement Π_2^c eines beliebigen Problems Π_2 liegt in $\mathcal{NP}$.

$\mathcal{NP}$ = co-$\mathcal{NP}$?

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Sei Π_1 ein $\mathcal{NP}$-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathcal{NP}$

Zeige: Komplement Π_2^c eines beliebigen Problems Π_2 liegt in $\mathcal{NP}$.

Π_1 ist $\mathcal{NP}$-vollständig: Es gibt Reduktion φ von Π_2 auf Π_1

φ ist auch Reduktion von Π_2^c auf Π_1^c

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$\mathcal{NP}$ = co-$\mathcal{NP}$?

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Sei Π_1 ein $\mathcal{NP}$-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathcal{NP}$

Zeige: Komplement Π_2^c eines beliebigen Problems Π_2 liegt in $\mathcal{NP}$.

Π_1 ist $\mathcal{NP}$-vollständig: Es gibt Reduktion φ von Π_2 auf Π_1

φ ist auch Reduktion von Π_2^c auf Π_1^c

Sei I_2^c eine Instanz von Π_2^c.

Transformiere die Instanz in eine Instanz von Π_1^c: $I_1^c = \varphi(I_2^c)$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

$\mathcal{NP}$ = co-$\mathcal{NP}$?

Zeigen Sie, dass falls das Komplement eines $\mathcal{NP}$-vollständigen Problems in $\mathcal{NP}$ liegt, dann gilt $\mathcal{NP}$=co-$\mathcal{NP}$

Sei Π_1 ein $\mathcal{NP}$-vollständiges Problem und sei Π_1^c das Komplement hierzu.

Annahme: $\Pi_1^c \in \mathcal{NP}$

Zeige: Komplement Π_2^c eines beliebigen Problems Π_2 liegt in $\mathcal{NP}$.

Π_1 ist $\mathcal{NP}$-vollständig: Es gibt Reduktion φ von Π_2 auf Π_1

φ ist auch Reduktion von Π_2^c auf Π_1^c

Sei I_2^c eine Instanz von Π_2^c.

Transformiere die Instanz in eine Instanz von Π_1^c: $I_1^c = \varphi(I_2^c)$.

Da Π_1^c in $\mathcal{NP}$ liegt, kann eine NTM $\mathcal{M}$ wie folgt auf I_1^c arbeiten.

1. $\mathcal{M}$ berechnet eine Lösung von I_1^c
2. $\mathcal{M}$ überprüft ob Ja-Instanz
3. Falls ja, dann auch I_2^c sonst ist I_2^c keine Ja-Instanz.