

# Theoretische Grundlagen der Informatik

## Übung

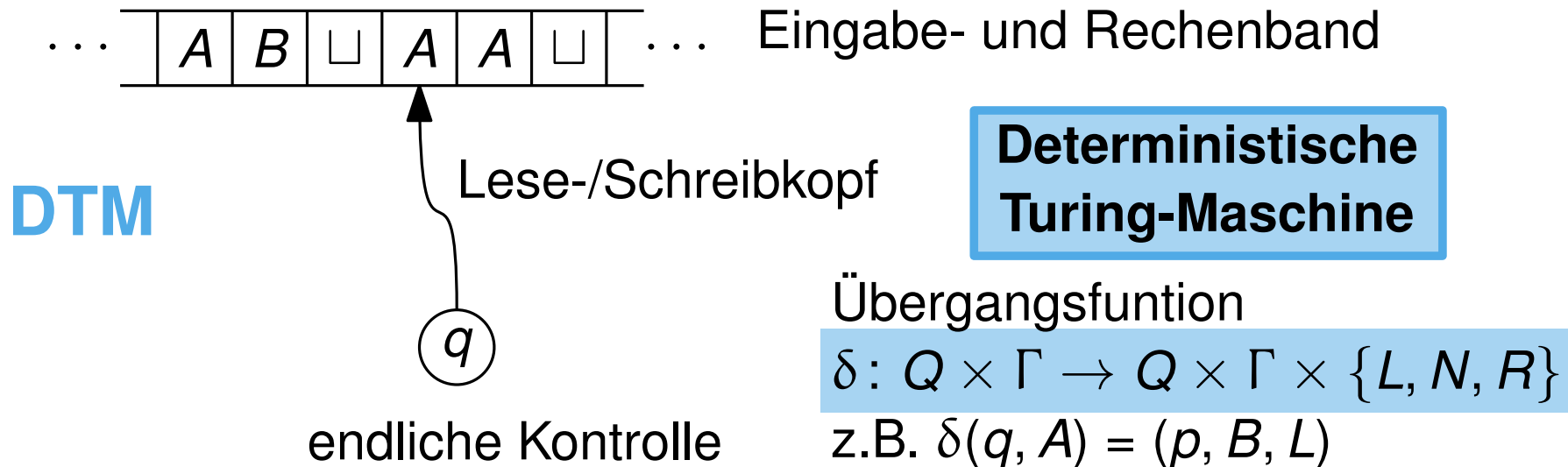
5. Übungstermin · 6. Dezember 2018  
Guido Brückner

INSTITUT FÜR THEORETISCHE INFORMATIK · LEHRSTUHL ALGORITHMIK

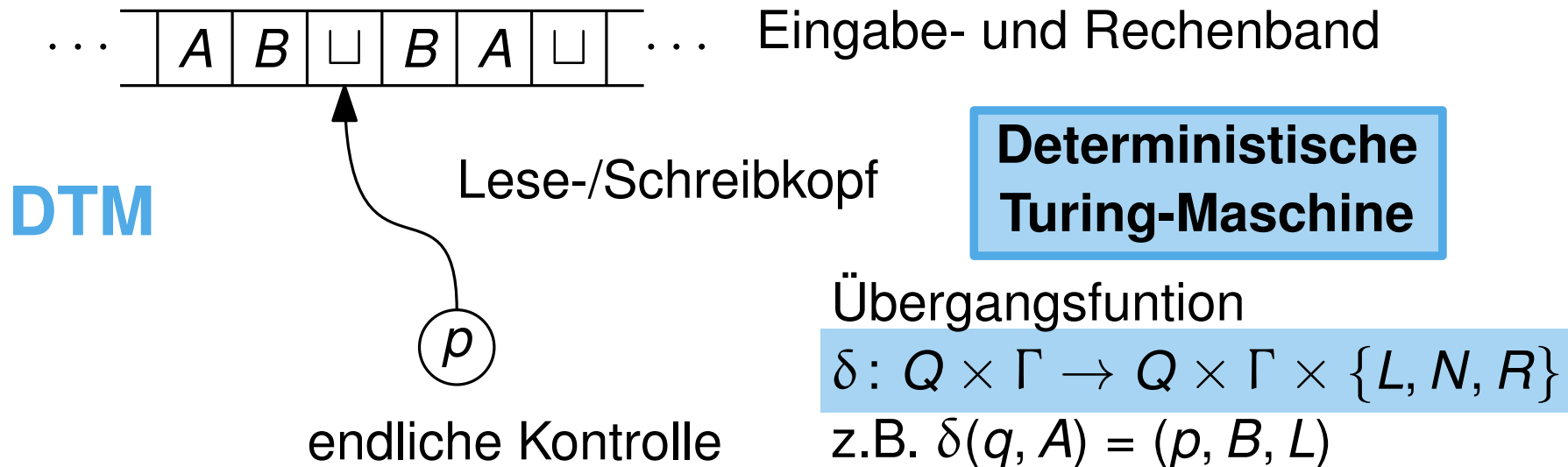
## Inhalt

- Nichtdeterministische Turing-Maschine(n)
- Polynomiale Transformation
- NP-vollständige Probleme

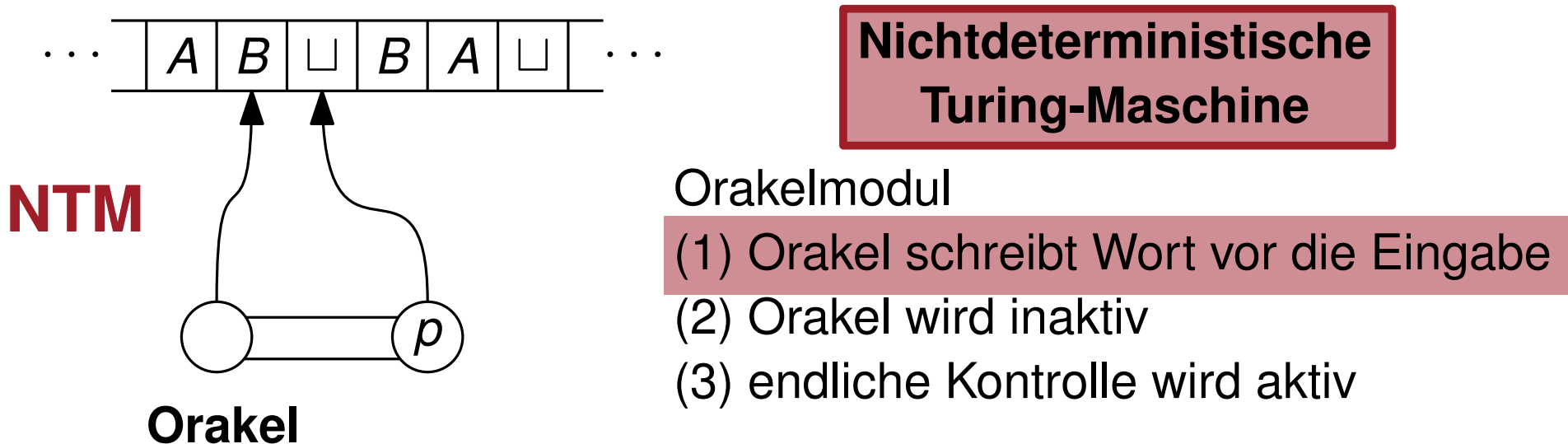
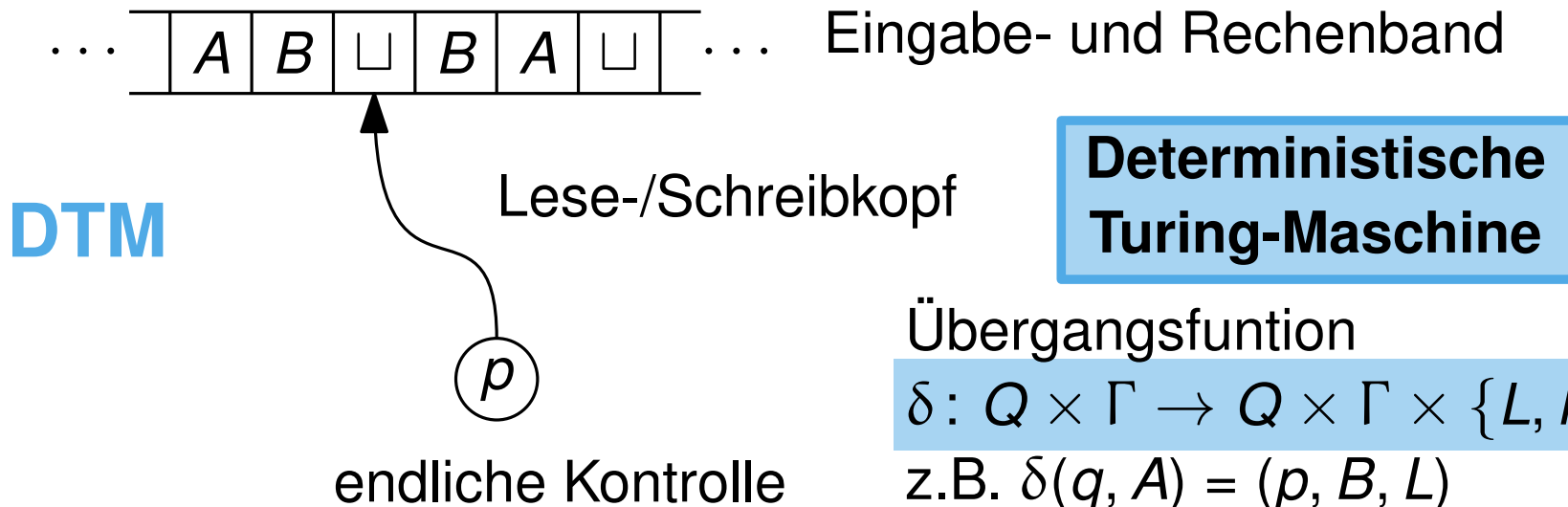
# Nichtdeterministische Turing-Maschine(n)



# Nichtdeterministische Turing-Maschine(n)



# Nichtdeterministische Turing-Maschine(n)



# Nichtdeterministische Turing-Maschine(n)



**A-NTM**



**Alternative nichtdeterministische Turing-Maschine**

Erweiterte Übergangsfunktion

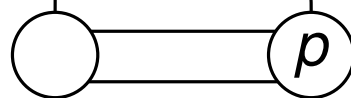
$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, N, R\}}$$

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

**Wahlmöglichkeiten**



**NTM**



**Orakel**

**Nichtdeterministische Turing-Maschine**

Orakelmodul

- (1) Orakel schreibt Wort vor die Eingabe
- (2) Orakel wird inaktiv
- (3) endliche Kontrolle wird aktiv

# Nichtdeterministische Turing-Maschine(n)



**Alternative nichtdeterministische Turing-Maschine**

**A-NTM**



Erweiterte Übergangsfunktion

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

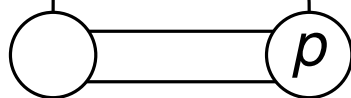
z.B.  $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$

**Wahlmöglichkeiten**



**Nichtdeterministische Turing-Maschine**

**NTM**



**Orakel**

Orakelmodul

- (1) Orakel schreibt Wort vor die Eingabe
- (2) Orakel wird inaktiv
- (3) endliche Kontrolle wird aktiv

# Nichtdeterministische Turing-Maschine(n)



**A-NTM**



**Alternative nichtdeterministische Turing-Maschine**

Erweiterte Übergangsfunktion

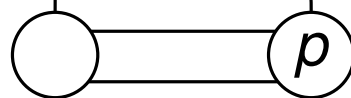
$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, N, R\}}$$

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

**Wahlmöglichkeiten**



**NTM**



**Orakel**

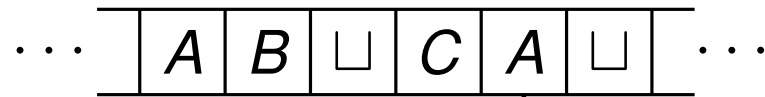
**Nichtdeterministische Turing-Maschine**

Orakelmodul

- (1) Orakel schreibt Wort vor die Eingabe
- (2) Orakel wird inaktiv
- (3) endliche Kontrolle wird aktiv



# Nichtdeterministische Turing-Maschine(n)



**A-NTM**



**Alternative nichtdeterministische Turing-Maschine**

Erweiterte Übergangsfunktion

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

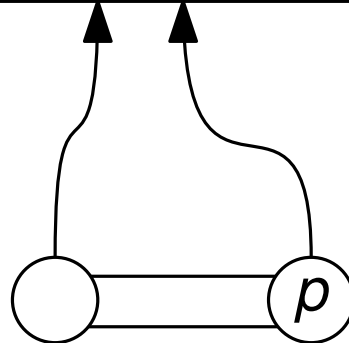
z.B.  $\delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$

**Wahlmöglichkeiten**



**Nichtdeterministische Turing-Maschine**

**NTM**



**Orakel**

Orakelmodul

- (1) Orakel schreibt Wort vor die Eingabe
- (2) Orakel wird inaktiv
- (3) endliche Kontrolle wird aktiv

# Nichtdeterministische Turing-Maschine(n)



**Alternative  
nichtdeterministische  
Turing-Maschine**

**A-NTM**



Erweiterte Übergangsfunktion

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

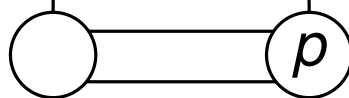
$$\text{z.B. } \delta(q, A) = \{(p, B, L), (r, C, R), (t, A, L)\}$$

**Wahlmöglichkeiten**



**Nichtdeterministische  
Turing-Maschine**

**NTM**



**Orakel**

Orakelmodul

- (1) Orakel schreibt Wort vor die Eingabe
- (2) Orakel wird inaktiv
- (3) endliche Kontrolle wird aktiv

# Nichtdeterministische Turing-Maschine(n)



**A-NTM**



**Wahlmöglichkeiten**

**Alternative  
nichtdeterministische  
Turing-Maschine**

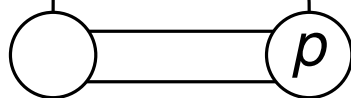
Erweiterte Übergangsfunktion

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

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



**NTM**



**Orakel**

**Nichtdeterministische  
Turing-Maschine**

Orakelmodul

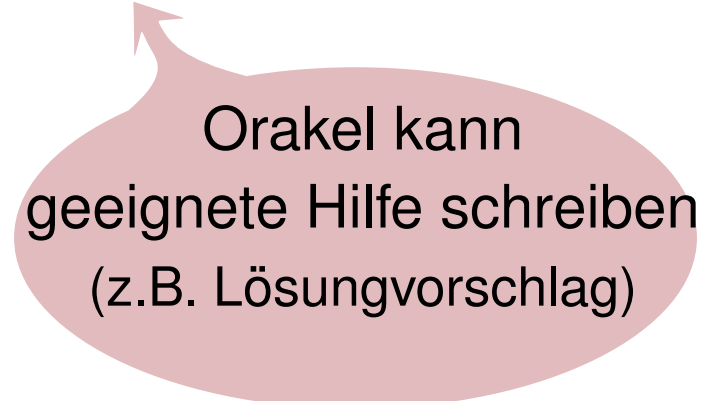
- (1) Orakel schreibt Wort vor die Eingabe
- (2) Orakel wird inaktiv
- (3) endliche Kontrolle wird aktiv

# Die Klassen NP und ANP

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

# Die Klassen NP und ANP

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



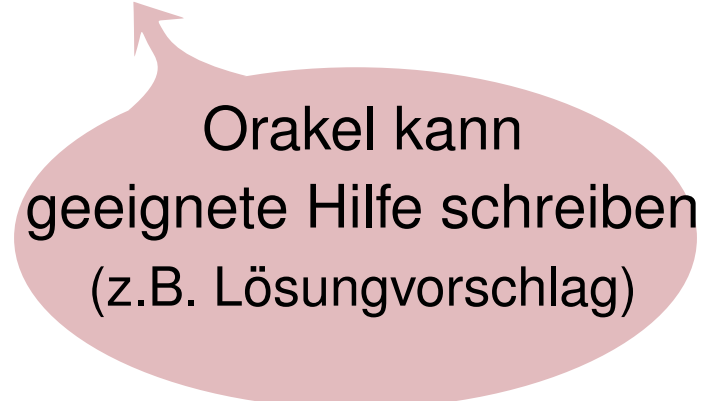
Orakel kann  
geeignete Hilfe schreiben  
(z.B. Lösungsvorschlag)

# Die Klassen NP und ANP

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

Die Klasse NP:

= alle Sprachen, die von einer **NTM** in **polynomialer Laufzeit** akzeptiert werden



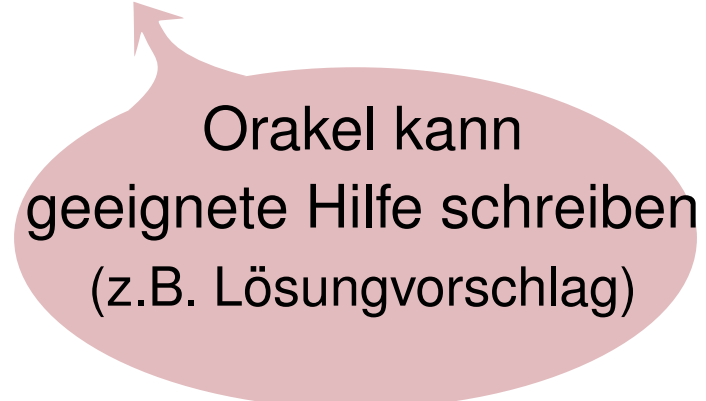
Orakel kann geeignete Hilfe schreiben (z.B. Lösungsvorschlag)

# Die Klassen NP und ANP

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

Die Klasse NP:

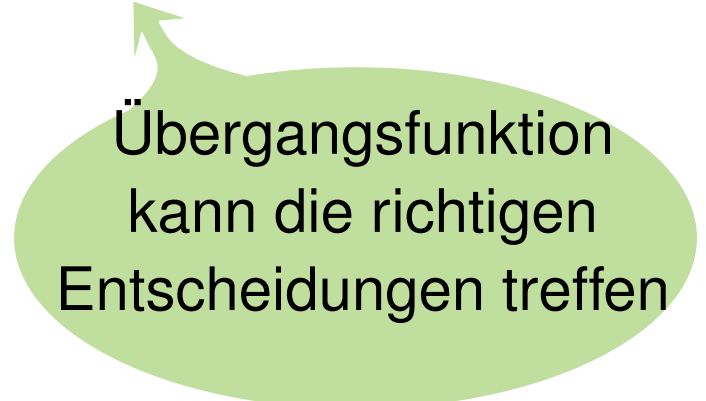
= alle Sprachen, die von einer **NTM** in polynomialer Laufzeit akzeptiert werden



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

---

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



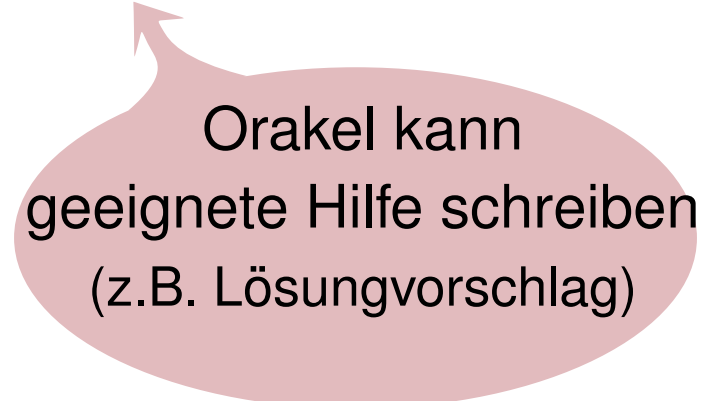
Übergangsfunktion kann die richtigen Entscheidungen treffen

# Die Klassen NP und ANP

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

Die Klasse NP:

= alle Sprachen, die von einer **NTM** in polynomialer Laufzeit akzeptiert werden



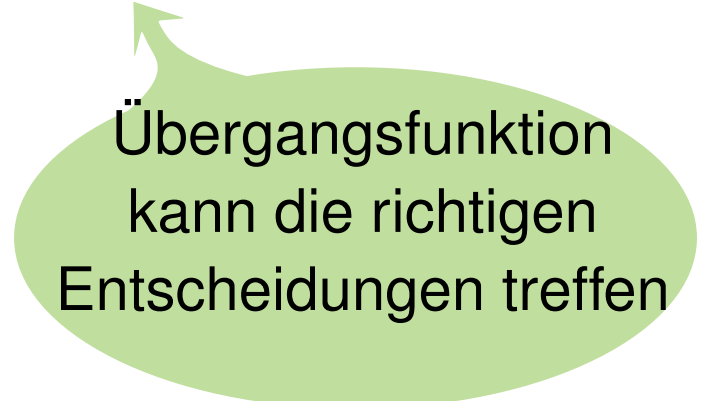
Orakel kann geeignete Hilfe schreiben (z.B. Lösungsvorschlag)

---

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

Die Klasse ANP:

= alle Sprachen, die von einer **A-NTM** in polynomialer Laufzeit akzeptiert werden



Übergangsfunktion kann die richtigen Entscheidungen treffen



# A-NTMs sind nicht mächtiger als NTMs

Behauptung:  $ANP \subseteq NP$

Wir müssen zeigen:

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

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

# A-NTMs sind nicht mächtiger als NTMs

Behauptung:  $ANP \subseteq NP$

Wir müssen zeigen:

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

- 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  $1, \dots, K \notin \Gamma_M$ .
- **Idee:** Orakel von  $M'$  schreibt Zahlenfolge der “richtigen Entscheidungen”

# A-NTMs sind nicht mächtiger als NTMs


**Behauptung:**  $ANP \subseteq NP$

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  $1, \dots, K \notin \Gamma_M$ .
- **Idee:** Orakel von  $M'$  schreibt Folge der “richtigen Entscheidungen”
- Definiere deterministischen Teil von  $M'$  als **2-Kopf TM**  $M'_{det}$  durch

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

1. Kopf liest Orakel-Vorgabe  
2. Kopf, endl. Kontrolle folgen  $\delta_M(q, A)$



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

# A-NTMs sind nicht mächtiger als NTMs

Behauptung:  $ANP \subseteq NP$

Wir müssen zeigen:

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

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

# A-NTMs sind nicht mächtiger als NTMs

Behauptung:  $ANP \subseteq NP$

Wir müssen zeigen:

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

- 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  $x$ .
- ⇒ Es gibt Übergänge\*, die  $M$  bei Eingabe  $x$  in  $q_j$  überführen.
  - ⇒ Es gibt Orakel-Hinweis der  $M'$  bei Eingabe  $x$  in  $q_j$  überführt.
  - ⇒  $M'$  akzeptiert Eingabe  $x$ .

\* aka. Entscheidungen bei Wahlmöglichkeiten

# A-NTMs sind nicht mächtiger als NTMs

Behauptung:  $ANP \subseteq NP$

Wir müssen zeigen:

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

- 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  $x$ .
- $\iff$  Es gibt Übergänge\*, die  $M$  bei Eingabe  $x$  in  $q_j$  überführen.
  - $\iff$  Es gibt Orakel-Hinweis der  $M'$  bei Eingabe  $x$  in  $q_j$  überführt.
  - $\iff$   $M'$  akzeptiert Eingabe  $x$ .

\* aka. Entscheidungen bei Wahlmöglichkeiten

# A-NTMs sind nicht mächtiger als NTMs

Behauptung:  $ANP \subseteq NP$

Wir müssen zeigen:

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

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-Schreibzeit + #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:  $ANP \subseteq NP$

Wir müssen zeigen:

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

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-Schreibzeit + #Schritte, um  $M'_{\text{det}}$  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  $ANP \subseteq NP$  gezeigt.



## Inhalt

- Nichtdeterministische Turing-Maschine(n)
- Polynomiale Transformation
- NP-vollständige Probleme

Eine **polynomiale Transformation** einer Sprache  $L_1 \subseteq \Sigma_1^*$  in eine Sprache  $L_2 \subseteq \Sigma_2^*$  ist eine Funktion  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  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  $L_2$ ).

# Polynomiale Transformation

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

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

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

# Polynomiale Transformation

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

$L_2$  = 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  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  definiert als

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

**Beispiel:**

$$f(1301) = aa$$

$$f(1000) = a$$

# Polynomiale Transformation

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

$L_2$  = 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  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  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.

# Polynomiale Transformation

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

$L_2$  = 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  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  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  $w \in \Sigma_1^*$  gilt:  $w \in L_1 \Leftrightarrow f(w) \in L_2$

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

# Komplexitätsklassen und Werkzeugkasten

Die Klasse NP:

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

Die Klasse P:

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

NP-schwere Probleme:

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

# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

Ist  $\Pi \in P$ ?

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

(2) Konstruiere **DTM**, die  $\Pi$  löst.



# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

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

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

(2) Konstruiere **DTM**, die  $\Pi$  löst.

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

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

(2) Konstruiere **NTM**, die  $\Pi$  löst.

# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

Ist  $\Pi \in P$ ?

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

(2) Konstruiere **DTM**, die  $\Pi$  löst.

Ist  $\Pi \in NP$ ?

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

(2) Konstruiere **NTM**, die  $\Pi$  löst.

Ist  $\Pi$  NP-schwer?

(1) Zeige  $\Pi' \propto \Pi$  für bekanntes  $\Pi'$  NP-schwer.

(2) Beweise ad hoc, dass  $\Pi$  NP-schwer.

# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

Ist  $\Pi \in P$ ?

**(1)** Zeige  $\Pi \propto \Pi'$  für bekanntes  $\Pi' \in P$ .

**(2)** Konstruiere **DTM**, die  $\Pi$  löst.

 gute Idee

Ist  $\Pi \in NP$ ?

**(1)** Zeige  $\Pi \propto \Pi'$  für bekanntes  $\Pi' \in NP$ .

**(2)** Konstruiere **NTM**, die  $\Pi$  löst.

 schlechte Idee

Ist  $\Pi$  NP-schwer?

**(1)** Zeige  $\Pi' \propto \Pi$  für bekanntes  $\Pi'$  NP-schwer.

**(2)** Beweise ad hoc, dass  $\Pi$  NP-schwer.

# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

Ist  $\Pi \in P$ ?

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

(2) Konstruiere **DTM**, die  $\Pi$  löst.

 gute Idee

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

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

(2) Konstruiere **NTM**, die  $\Pi$  löst.

 schlechte Idee

Ist  $\Pi$  NP-schwer?

(1) Zeige  $\Pi' \propto \Pi$  für bekanntes  $\Pi'$  NP-schwer.

(2) Beweise ad hoc, dass  $\Pi$  NP-schwer.

# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

Ist  $\Pi \in P$ ?

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

(2) Konstruiere **DTM**, die  $\Pi$  löst.

 gute Idee

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


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

(2) Konstruiere **NTM**, die  $\Pi$  löst.

 schlechte Idee

Ist  $\Pi$  NP-schwer?

(1) Zeige  $\Pi' \propto \Pi$  für bekanntes  $\Pi'$  NP-schwer.

(2)  Beweise ad hoc, dass  $\Pi$  NP-schwer.

# 2SAT $\in$ 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 P liegt.

# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

Ist  $\Pi \in P$ ?

**(1)** Zeige  $\Pi \propto \Pi'$  für bekanntes  $\Pi' \in P$ .

**(2)** Konstruiere **DTM**, die  $\Pi$  löst.

 gute Idee

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

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

**(2)** Konstruiere **NTM**, die  $\Pi$  löst.

 schlechte Idee

Ist  $\Pi$  NP-schwer?

**(1)** Zeige  $\Pi' \propto \Pi$  für bekanntes  $\Pi'$  NP-schwer.

**(2)** Beweise ad hoc, dass  $\Pi$  NP-schwer.



# 2SAT $\in$ P

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

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

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

**Es gilt:**

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

# 2SAT $\in$ P

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

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

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

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

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

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



5 min Zeit



**Es gilt:**

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

# 2SAT $\in P$

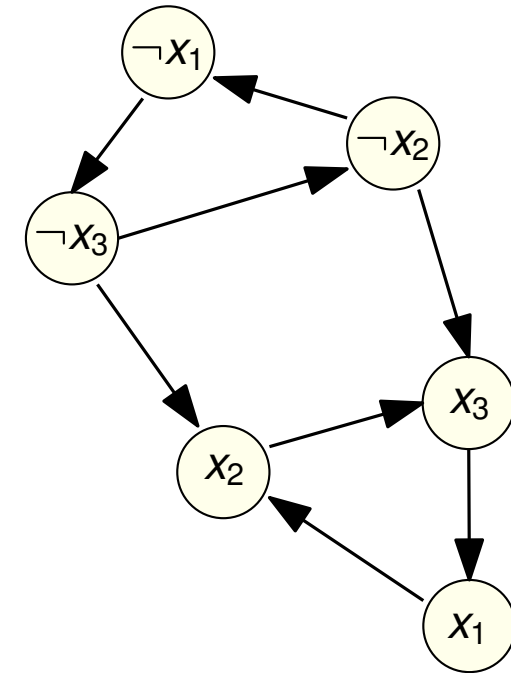
Konstruiere gerichteten Graphen  $G = (V, E)$

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

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

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

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



**Es gilt:**

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

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

# 2SAT $\in P$

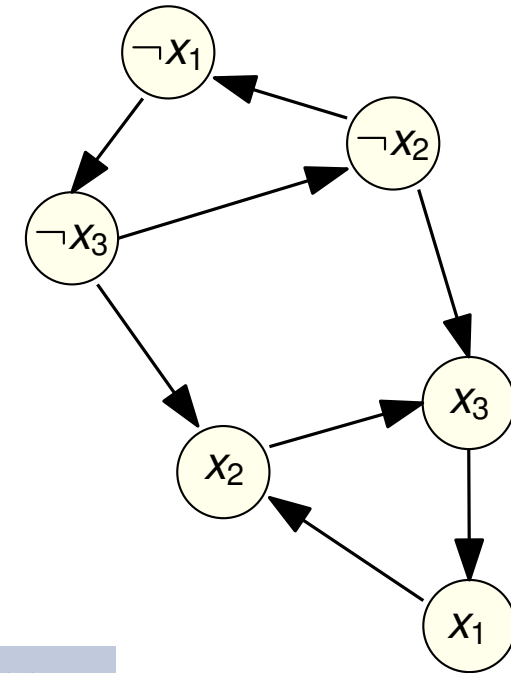
Konstruiere gerichteten Graphen  $G = (V, E)$

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

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

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

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



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

**Es gilt:**

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

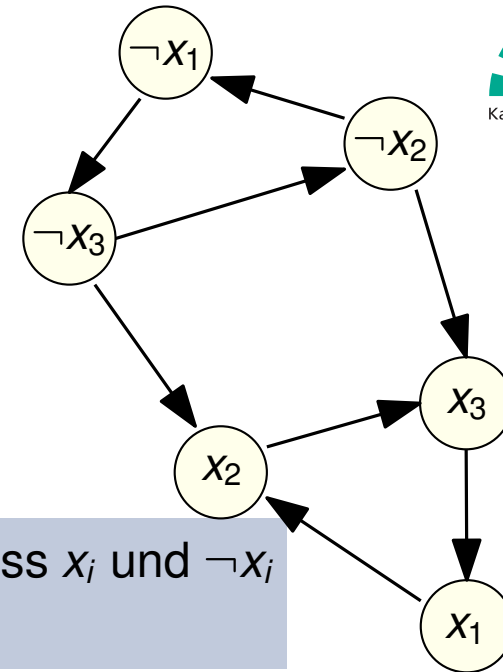
**Beispiel:**  $\neg x_1 \vee x_2$   $\neg x_2 \vee x_3$   $x_1 \vee \neg x_3$   $x_2 \vee x_3$

# 2SAT $\in P$

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

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

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



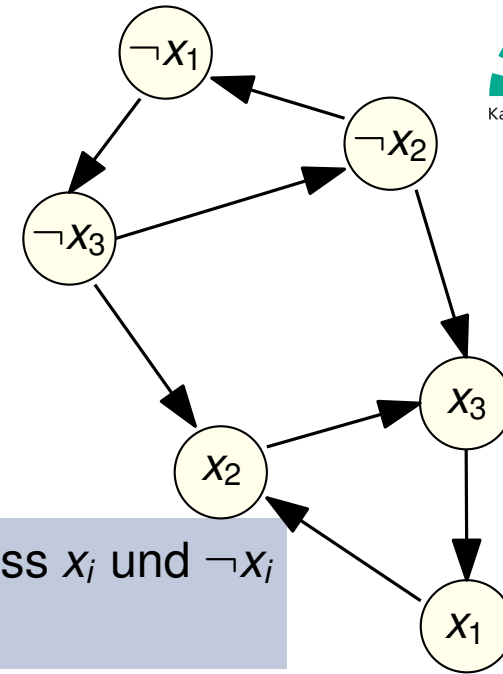
**Beispiel:**  $\neg x_1 \vee x_2$   $\neg x_2 \vee x_3$   $x_1 \vee \neg x_3$   $x_2 \vee x_3$

# 2SAT $\in$ P

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

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

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



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

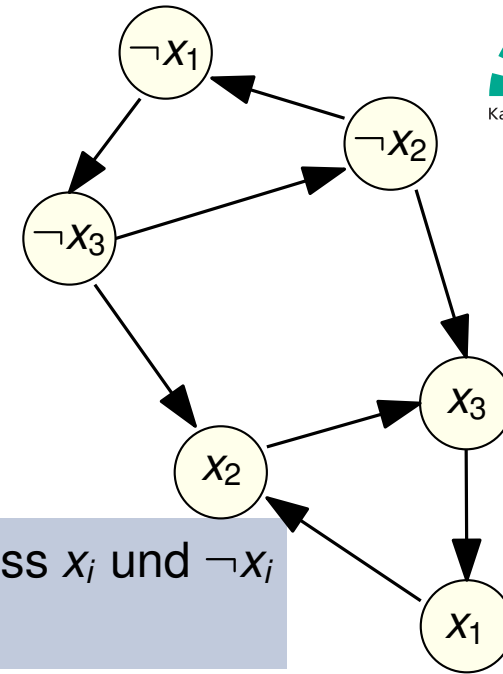
**Beispiel:**  $\neg x_1 \vee x_2 \quad \neg x_2 \vee x_3 \quad x_1 \vee \neg x_3 \quad x_2 \vee x_3$

# 2SAT $\in P$

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

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

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



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

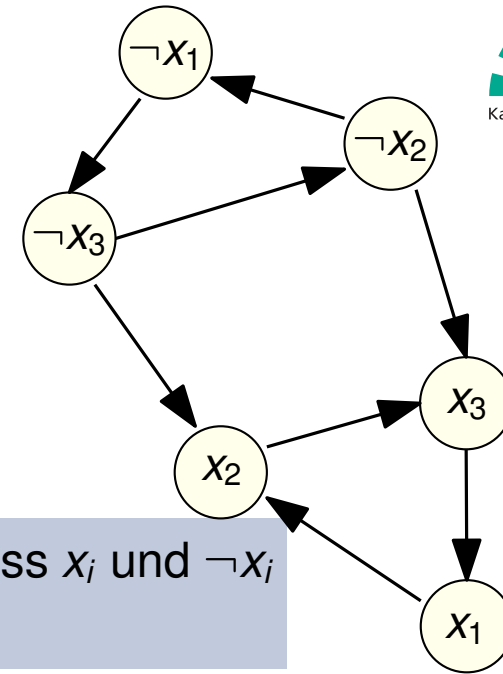
1. Es gibt gerichteten Pfad  $x_i, \dots, \neg x_i$ .  $\rightarrow$  Es gilt Implikation  $x_i \Rightarrow \neg x_i$
2. Es gibt gerichteten Pfad  $\neg x_i, \dots, x_i$ .  $\rightarrow$  Es gilt Implikation  $\neg x_i \Rightarrow x_i$

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

# 2SAT $\in P$

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

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



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

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

1. Es gibt gerichteten Pfad  $x_i, \dots, \neg x_i$ .  $\rightarrow$  Es gilt Implikation  $x_i \Rightarrow \neg x_i$
2. Es gibt gerichteten Pfad  $\neg x_i, \dots, x_i$ .  $\rightarrow$  Es gilt Implikation  $\neg x_i \Rightarrow x_i$



$C$  erfüllbar

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

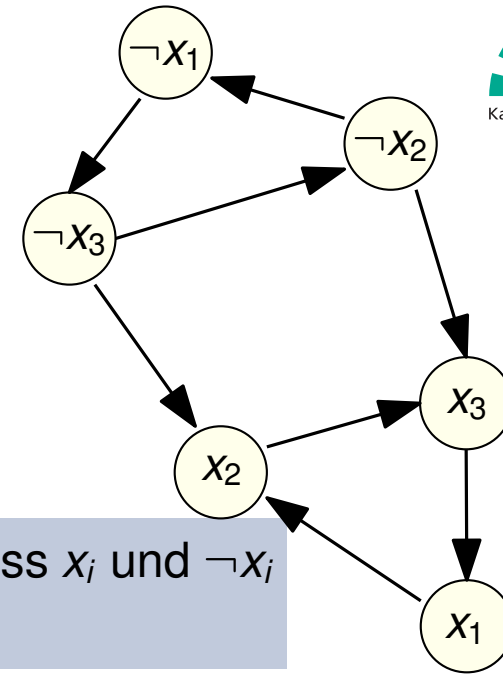


# 2SAT $\in P$

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

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

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



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

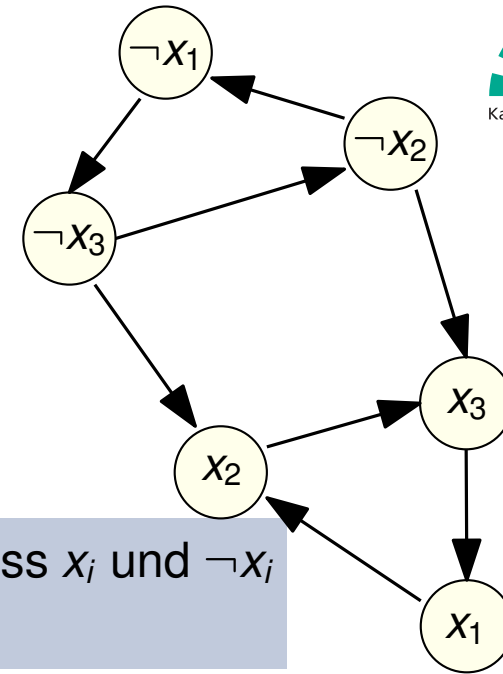
**Beispiel:**  $\neg x_1 \vee x_2$   $\neg x_2 \vee x_3$   $x_1 \vee \neg x_3$   $x_2 \vee x_3$

# 2SAT $\in P$

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

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

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



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

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

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

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

# NP-Vollständigkeit

Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:

- $L \in \text{NP}$  und
- für alle  $L' \in \text{NP}$  gilt  $L' \leq L$ .

## Aussage aus Vorlesung:

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

**Schema für Beweis, dass Problem  $\Pi$  NP-vollständig ist.**

Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:

- $L \in \text{NP}$  und
- für alle  $L' \in \text{NP}$  gilt  $L' \leq L$ .

## Aussage aus Vorlesung:

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

## Schema für Beweis, dass Problem $\Pi$ NP-vollständig ist.

**1. Schritt:** Zeige, dass  $\Pi$  in NP liegt.

**2. Schritt:** Zeige, dass es NP-vollständiges Problem  $\Pi'$  gibt mit  $\Pi' \leq \Pi$

Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:

- $L \in \text{NP}$  und
- für alle  $L' \in \text{NP}$  gilt  $L' \leq L$ .

## Aussage aus Vorlesung:

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

## Schema für Beweis, dass Problem $\Pi$ NP-vollständig ist.

**1. Schritt:** Zeige, dass  $\Pi$  in NP liegt.

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

**2. Schritt:** Zeige, dass es NP-vollständiges Problem  $\Pi'$  gibt mit  $\Pi' \leq \Pi$

Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:

- $L \in \text{NP}$  und
- für alle  $L' \in \text{NP}$  gilt  $L' \leq L$ .

## Aussage aus Vorlesung:

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

## Schema für Beweis, dass Problem $\Pi$ NP-vollständig ist.

**1. Schritt:** Zeige, dass  $\Pi$  in NP liegt.

- Orakel rät mögliche Lösung  $\mathcal{L}$  für gegebene Instanz
- 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 NP-vollständiges Problem  $\Pi'$  gibt mit  $\Pi' \leq \Pi$

Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:

- $L \in \text{NP}$  und
- für alle  $L' \in \text{NP}$  gilt  $L' \leq L$ .

## Aussage aus Vorlesung:

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

## Schema für Beweis, dass Problem $\Pi$ NP-vollständig ist.

**1. Schritt:** Zeige, dass  $\Pi$  in NP liegt.

**2. Schritt:** Zeige, dass es NP-vollständiges Problem  $\Pi'$  gibt mit  $\Pi' \leq \Pi$



Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:

- $L \in \text{NP}$  und
- für alle  $L' \in \text{NP}$  gilt  $L' \leq L$ .

## Aussage aus Vorlesung:

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

## Schema für Beweis, dass Problem $\Pi$ NP-vollständig ist.

**1. Schritt:** Zeige, dass  $\Pi$  in NP liegt.

**2. Schritt:** Zeige, dass es NP-vollständiges Problem  $\Pi'$  gibt mit  $\Pi' \leq \Pi$

a) Gebe polynomielle Reduktion  $\leq$  an, so dass  $\Pi' \leq \Pi$ .

Zeige, dass  $\leq$  tatsächlich polynomiell ist: z.B. beweise Laufzeit im  $\mathcal{O}$ -Kalkül.

Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:

- $L \in \text{NP}$  und
- für alle  $L' \in \text{NP}$  gilt  $L' \leq L$ .

## Aussage aus Vorlesung:

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

## Schema für Beweis, dass Problem $\Pi$ NP-vollständig ist.

**1. Schritt:** Zeige, dass  $\Pi$  in NP liegt.

**2. Schritt:** Zeige, dass es NP-vollständiges Problem  $\Pi'$  gibt mit  $\Pi' \leq \Pi$

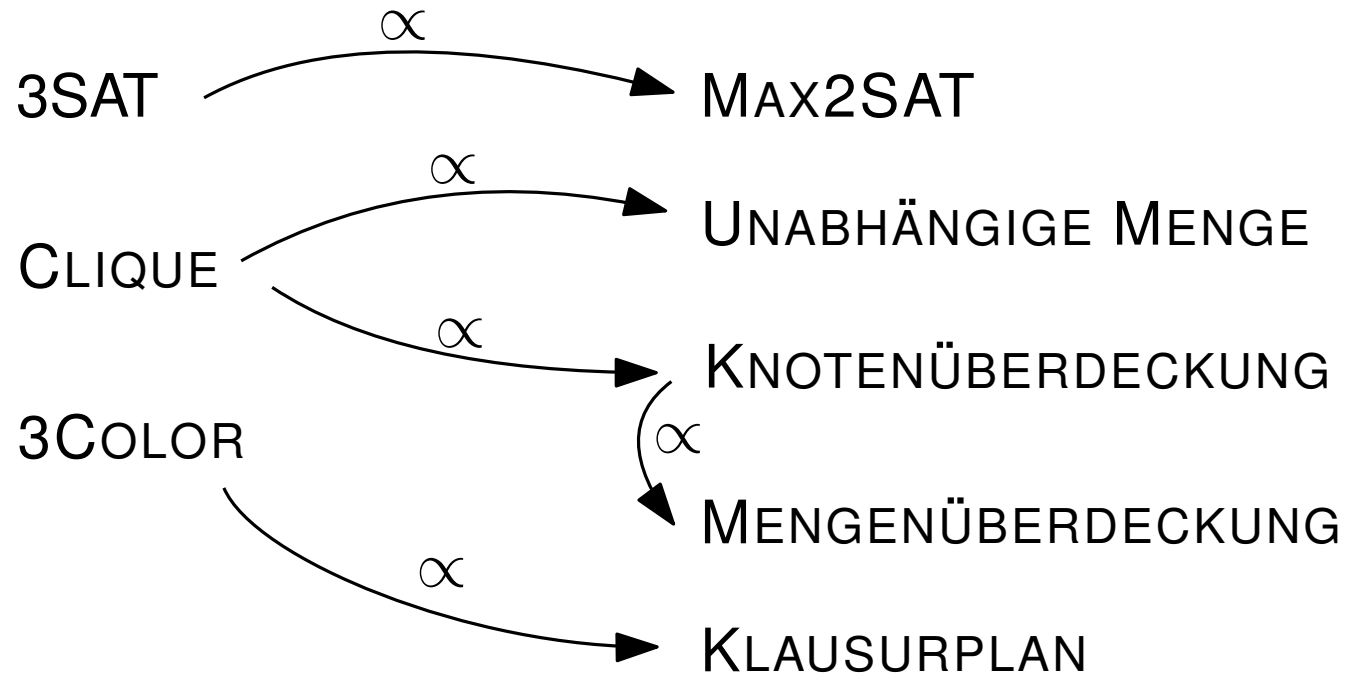
a) Gebe polynomielle Reduktion  $\leq$  an, so dass  $\Pi' \leq \Pi$ .

Zeige, dass  $\leq$  tatsächlich polynomiell ist: z.B. beweise Laufzeit im  $\mathcal{O}$ -Kalkül.

b) Sei  $I' \in \Pi'$  beliebige Instanz und sei  $I \in \Pi$  die Transformation von  $I'$  bzgl.  $\leq$ .

**Zeige:**  $I'$  ist eine *Ja*-Instanz von  $\Pi'$  **genau dann wenn**  $I$  ist eine *Ja*-Instanz von  $\Pi$ .

# Einige NP-vollständige Probleme



# MAX2SAT ist NP-vollständig

# MAX2SAT

## Problem MAX2SAT:

**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 NP-vollständig ist.

## Problem MAX2SAT:

**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 NP-vollständig ist.

**1. Schritt:** MAX2SAT liegt in NP.

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

# MAX2SAT

## Problem MAX2SAT:

**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 NP-vollständig ist.

**2. Schritt:** Reduktion von 3SAT auf MAX2SAT

# MAX2SAT

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

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

abbildet. Dabei ist  $w_c$  eine neu eingeführte boolesche Variable.



# MAX2SAT

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

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

abbildet. Dabei ist  $w_c$  eine neu eingeführte boolesche Variable.

i. Zeigen Sie, dass  $F_c$  nicht erfüllbar ist.

# MAX2SAT

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

$$F_c = \{ \underbrace{x, y, z, w_c}_I, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_II, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_III \},$$

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.

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

$$F_c = \{ \underbrace{x, y, z, w_c}_I, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_II, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_III \},$$

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.

# MAX2SAT

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

$$F_c = \{ \underbrace{x, y, z, w_c}_I, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_II, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_III \},$$

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

$$F_c = \{ \underbrace{x, y, z, w_c}_I, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_II, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_III \},$$

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		

# MAX2SAT

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

$$F_c = \{ \underbrace{x, y, z, w_c}_I, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_II, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_III \},$$

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		

# MAX2SAT

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

$$F_c = \{ \underbrace{x, y, z, w_c}_I, \underbrace{\neg x \vee \neg y, \neg y \vee \neg z, \neg x \vee \neg z}_II, \underbrace{x \vee \neg w_c, y \vee \neg w_c, z \vee \neg w_c}_III \},$$

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

# MAX2SAT

## Problem MAX2SAT:

**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 NP-vollständig ist.



## Problem MAX2SAT:

**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 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.

## Problem MAX2SAT:

**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 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?

## Problem MAX2SAT:

**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 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  $7m$ .

## Problem MAX2SAT:

**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 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 Wahrheitbelegung für  $I'$  gibt, die  $7k$  Klauseln erfüllt.

**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

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

⇒ 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

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

⇒ 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

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

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

⇒ 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

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

Pro Menge  $F_c$  sind maximal 7 Klauseln gleichzeitig erfüllt



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

⇒ 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

⇐ 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.

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

⇒ 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

⇐ 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.

---

⇒ In jeder Menge  $F_c$  sind 7 Klauseln erfüllt

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

⇒ 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

⇐ 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.

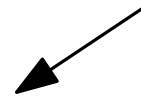
---

⇒ In jeder Menge  $F_c$  sind 7 Klauseln erfüllt

Aus   folgt, dass jede Klausel  $c$  von  $I$  ebenfalls erfüllt.

**Behauptung:** MAX2SAT ist NP-vollständig.

## Zusammenfassung des Beweises



MAX2SAT ist in NP



Konstruierte **NTM** löst Max2SAT



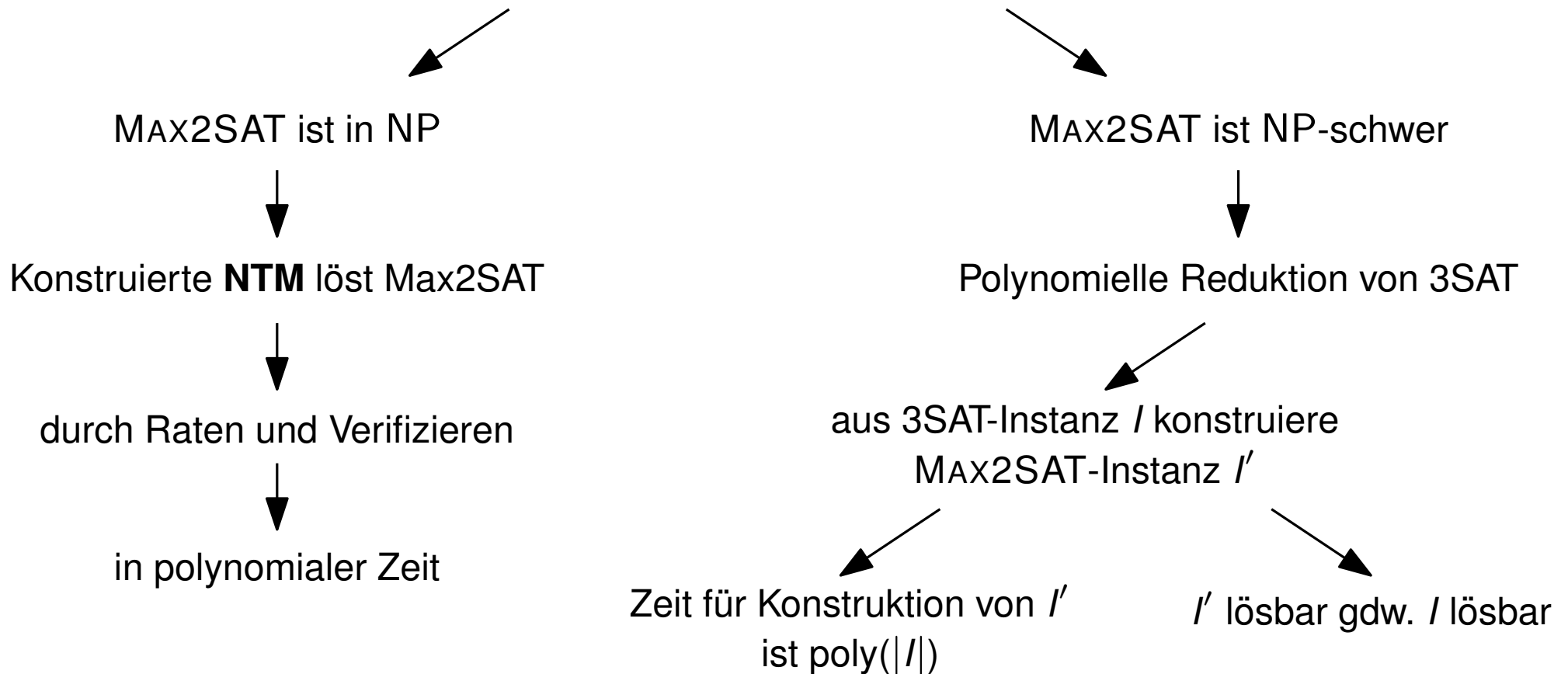
durch Raten und Verifizieren



in polynomialer Zeit

**Behauptung:** MAX2SAT ist NP-vollständig.

## Zusammenfassung des Beweises



# Komplexitätsklassen und Werkzeugkasten

Die Klasse **NP**:

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

Die Klasse **P**:

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

**NP-schwere** Probleme:

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

Ist  $\Pi \in P$ ?

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

(2) Konstruiere **DTM**, die  $\Pi$  löst.

 gute Idee

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


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

(2) Konstruiere **NTM**, die  $\Pi$  löst.

 schlechte Idee

Ist  $\Pi$  NP-schwer?

(1) Zeige  $\Pi' \propto \Pi$  für bekanntes  $\Pi'$  NP-schwer.

(2)  Beweise ad hoc, dass  $\Pi$  NP-schwer.

# UNABHÄNGIGE MENGE ist NP-vollständig

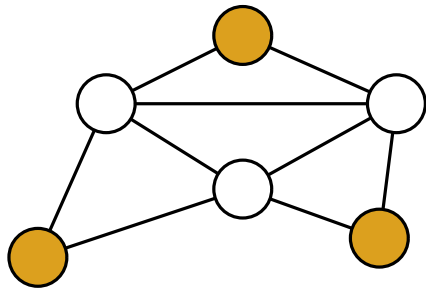
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

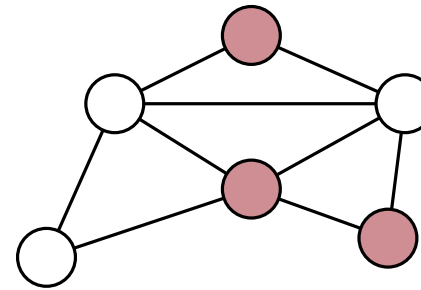
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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



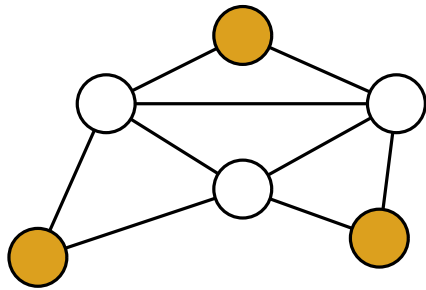
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

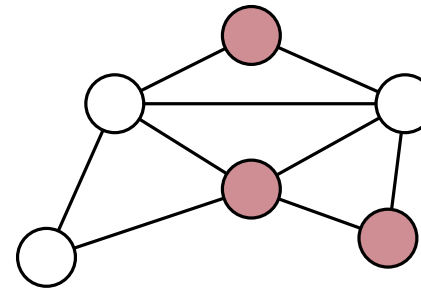
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP.

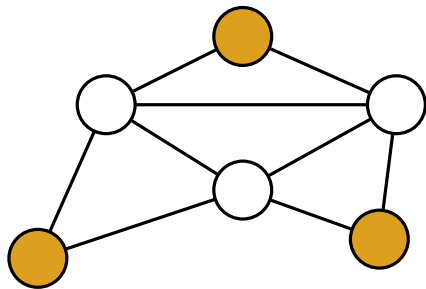
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

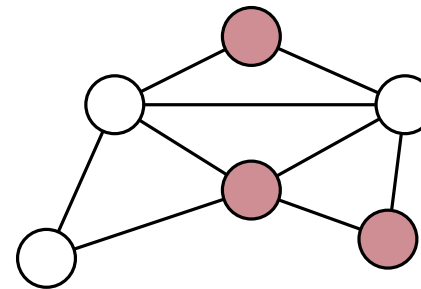
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 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  $\mathcal{O}(n^2)$

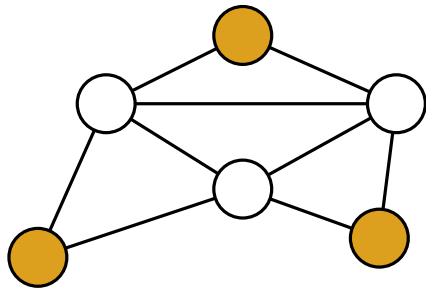
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

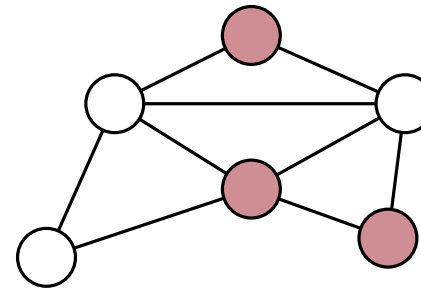
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

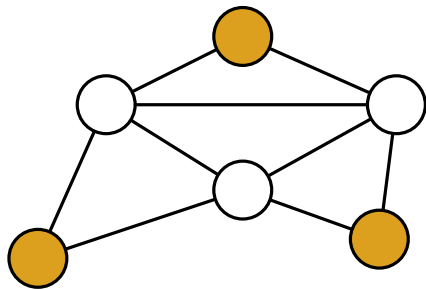
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

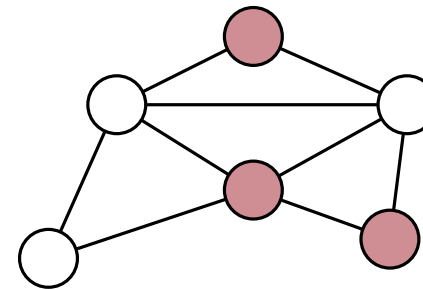
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 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.

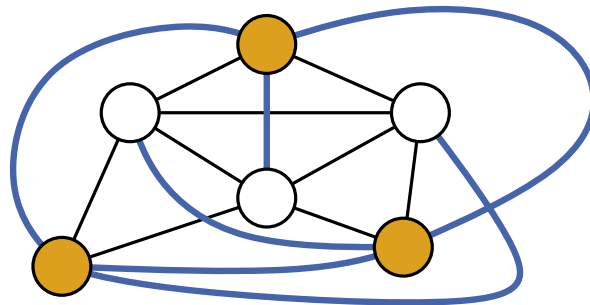
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

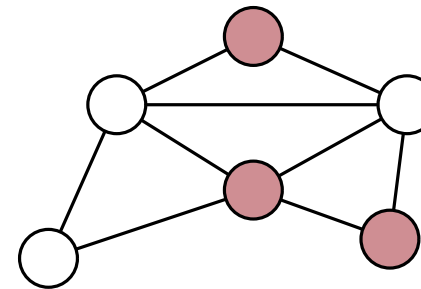
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

CLIQUE  $\propto$  UNABHÄNGIGE MENGE

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

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

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

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

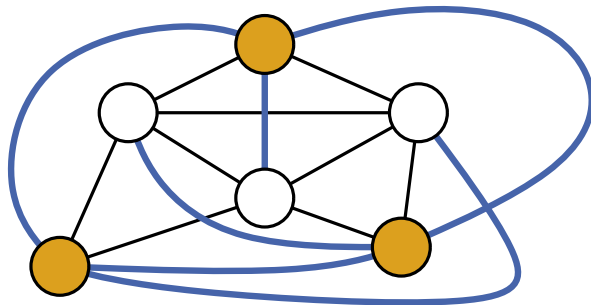
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

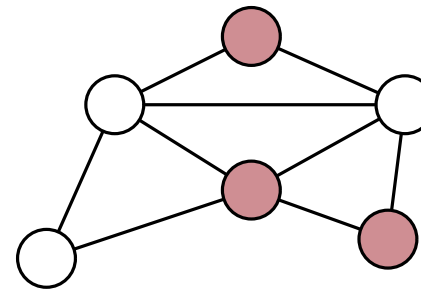
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

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

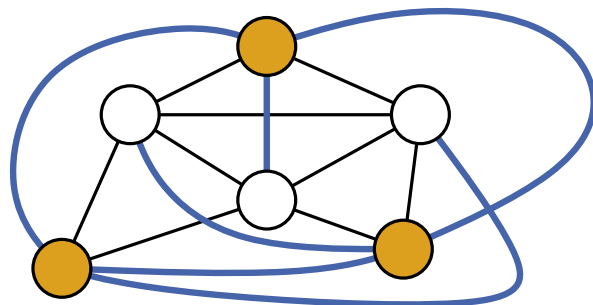
# Unabhängige Menge

## Problem UNABHÄNGIGE MENGE:

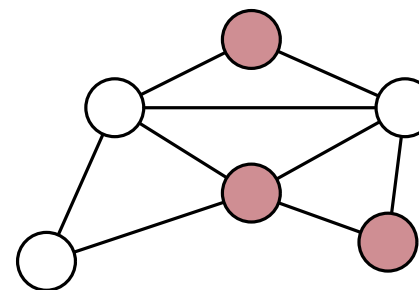
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

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

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

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

# KNOTENÜBERDECKUNG ist NP-vollständig



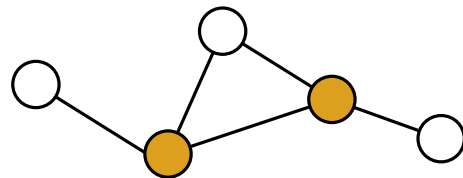
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

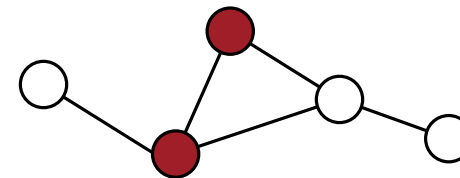
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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

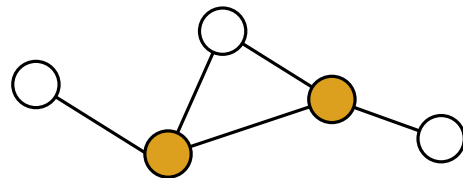
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

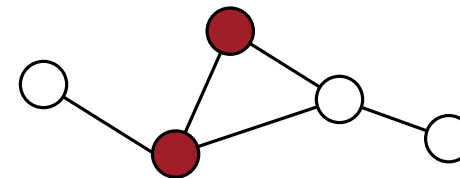
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP

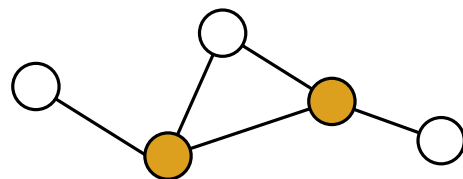
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

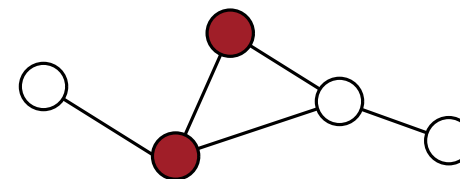
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 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  $\mathcal{O}(n^3)$

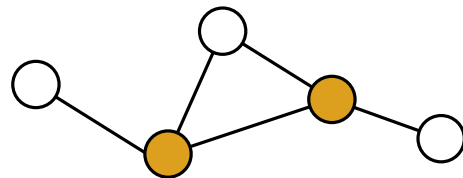
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

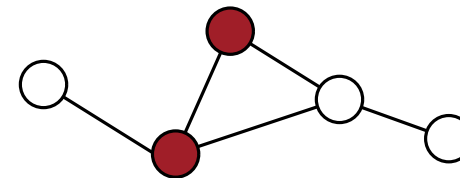
**Frage:** Gibt es eine Knotenüberdeckung  $V' \subseteq V$  mit  $|V'| \leq k$ ?

**Hinweis:**  $V' \subseteq V$  heißt *Knotenüberdeck.*, falls für alle  $\{u, v\} \in E$  gilt  $u \in V'$  oder  $v \in V'$ .



Knotenüberdeckung

**2. Schritt:** KNOTENÜBERDECKUNG ist NP-schwer.



Keine Knotenüberdeckung

CLIQUE  $\propto$  KNOTENÜBERDECKUNG

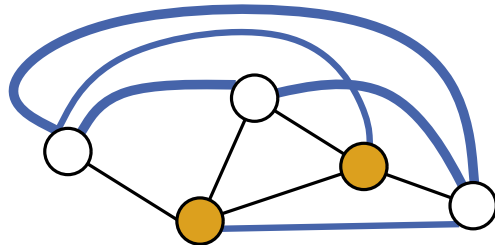
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

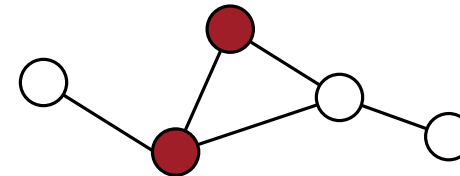
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

CLIQUE  $\propto$  KNOTENÜBERDECKUNG

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

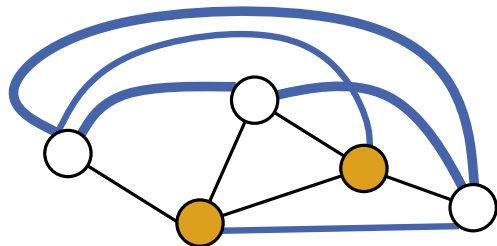
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

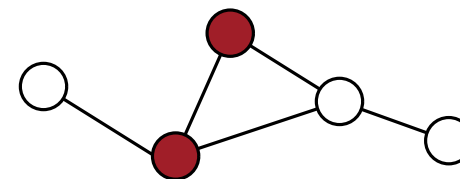
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

CLIQUE  $\propto$  KNOTENÜBERDECKUNG

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

## Beweis:

" $\Rightarrow$ " Angenommen es gibt  $\{u, v\} \in \bar{E}$ , 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  $G$



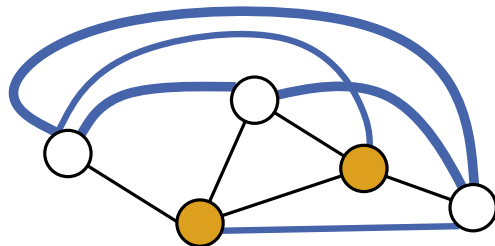
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

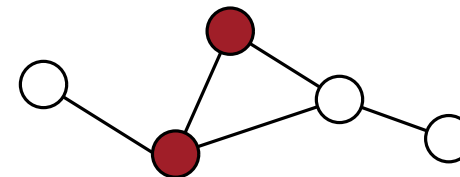
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

CLIQUE  $\propto$  KNOTENÜBERDECKUNG

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

## Beweis:

" $\Rightarrow$ " Angenommen es gibt  $\{u, v\} \in \bar{E}$ , 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.  $\bar{G}$

" $\Leftarrow$ " Seien  $u, v \in V \setminus V'$ . Es gibt keine Kante  $\{u, v\} \in \bar{E}$ , sonst ist  $V'$  keine Knotenüb.

Damit gibt es die Kante  $\{u, v\}$  in  $G$ .

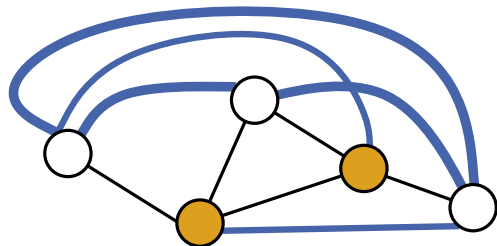
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

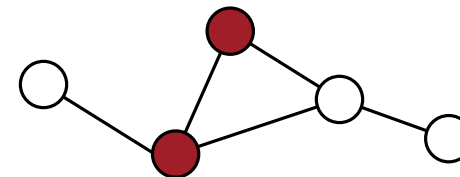
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 NP-schwer.

CLIQUE  $\propto$  KNOTENÜBERDECKUNG

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

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

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

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

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



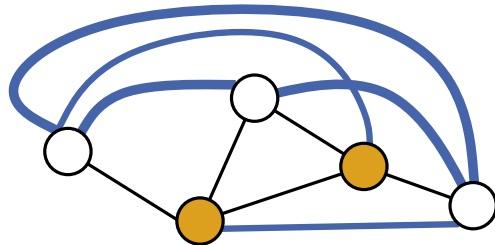
# Knotenüberdeckung

## Problem KNOTENÜBERDECKUNG:

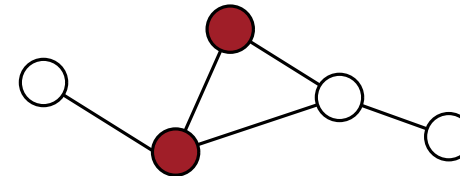
**Gegeben:** Ungerichteter Graph  $G = (V, E)$  und Zahl  $k \in \mathbb{N}$ .

**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 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  $\overline{G}$

# MENGENÜBERDECKUNG ist NP-vollständig

# Mengenüberdeckung

## Problem MENGENÜBERDECKUNG:

**Gegeben:** Universum  $\mathcal{U} = \{x_1, \dots, x_m\}$ , Teilmengen  $S_1, \dots, S_n \subseteq \mathcal{U}$  und Zahl  $k \in \mathbb{N}$ .

**Frage:** Gibt es eine Menge  $C \subseteq \{1, \dots, n\}$  mit  $|C| \leq k$ , sodass  $\bigcup_{i \in C} S_i = \mathcal{U}$ ?

*Hinweis:* Verwenden Sie das Problem KNOTENÜBERDECKUNG für die Reduktion.

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

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

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

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

## Problem MENGENÜBERDECKUNG:

**Gegeben:** Universum  $\mathcal{U} = \{x_1, \dots, x_m\}$ , Teilmengen  $S_1, \dots, S_n \subseteq \mathcal{U}$  und Zahl  $k \in \mathbb{N}$ .

**Frage:** Gibt es eine Menge  $C \subseteq \{1, \dots, n\}$  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 NP.

## Problem MENGENÜBERDECKUNG:

**Gegeben:** Universum  $\mathcal{U} = \{x_1, \dots, x_m\}$ , Teilmengen  $S_1, \dots, S_n \subseteq \mathcal{U}$  und Zahl  $k \in \mathbb{N}$ .

**Frage:** Gibt es eine Menge  $C \subseteq \{1, \dots, n\}$  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 NP.

- Orakel rät Zahlen  $i_1, \dots, i_\ell$ .
- 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.

# Mengenüberdeckung

## Problem MENGENÜBERDECKUNG:

**Gegeben:** Universum  $\mathcal{U} = \{x_1, \dots, x_m\}$ , Teilmengen  $S_1, \dots, S_n \subseteq \mathcal{U}$  und Zahl  $k \in \mathbb{N}$ .

**Frage:** Gibt es eine Menge  $C \subseteq \{1, \dots, n\}$  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 NP-schwer.

## Problem MENGENÜBERDECKUNG:

**Gegeben:** Universum  $\mathcal{U} = \{x_1, \dots, x_m\}$ , Teilmengen  $S_1, \dots, S_n \subseteq \mathcal{U}$  und Zahl  $k \in \mathbb{N}$ .

**Frage:** Gibt es eine Menge  $C \subseteq \{1, \dots, n\}$  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 NP-schwer.

**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

## Problem MENGENÜBERDECKUNG:

**Gegeben:** Universum  $\mathcal{U} = \{x_1, \dots, x_m\}$ , Teilmengen  $S_1, \dots, S_n \subseteq \mathcal{U}$  und Zahl  $k \in \mathbb{N}$ .

**Frage:** Gibt es eine Menge  $C \subseteq \{1, \dots, n\}$  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 NP-schwer.

## Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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



**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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

**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

1. Sei  $\mathcal{U} = E$
2. Def.  $n$  Teilmengen  $S_1, \dots, S_n$  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  $\ell = |V'| \leq k$

**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

1. Sei  $\mathcal{U} = E$
2. Def.  $n$  Teilmengen  $S_1, \dots, S_n$  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:**  $\bigcup_{i \in C} S_i = \mathcal{U}$

**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

1. Sei  $\mathcal{U} = E$
2. Def.  $n$  Teilmengen  $S_1, \dots, S_n$  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.

## Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

1. Sei  $\mathcal{U} = E$
2. Def.  $n$  Teilmengen  $S_1, \dots, S_n$  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  $e$ .

**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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

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

**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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

” $\Leftarrow$ ” Es gibt Menge  $C \subseteq \{1, \dots, n\}$  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  $v_i$ .

**Reduktion:** KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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

” $\Leftarrow$ ” Es gibt Menge  $C \subseteq \{1, \dots, n\}$  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 durch  $C$ , es gilt  $|V'| \leq m = k$ .



## Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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

” $\Leftarrow$ ” Es gibt Menge  $C \subseteq \{1, \dots, n\}$  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 durch  $C$ , es gilt  $|V'| \leq m = k$ .

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

## Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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

” $\Leftarrow$ ” Es gibt Menge  $C \subseteq \{1, \dots, n\}$  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 durch  $C$ , 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  $e \in S_i$ .

## Reduktion: KNOTENÜBERDECKUNG auf MENGENÜBERDECKUNG

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

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

” $\Leftarrow$ ” Es gibt Menge  $C \subseteq \{1, \dots, n\}$  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 durch  $C$ , es gilt  $|V'| \leq m = k$ .

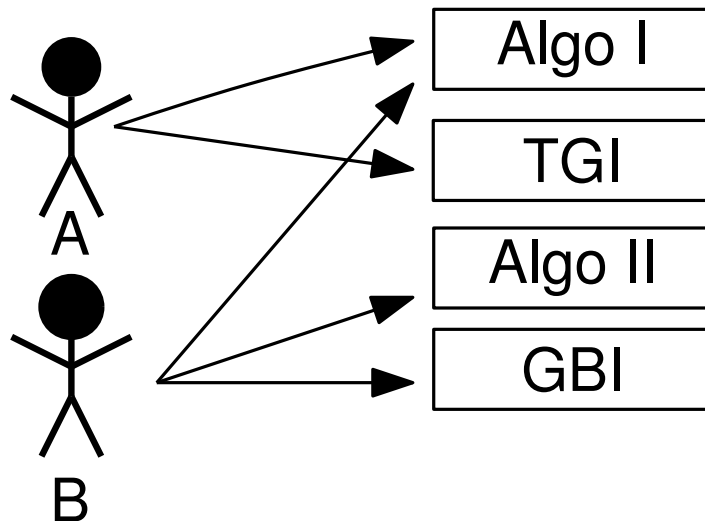
Betrachte beliebige Kante  $e \in \mathcal{U}$ .

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

# KLAUSURPLAN ist NP-vollständig

## Beispiel:



	Montag	Dienstag
1. Block		
2. Block		
3. Block		
4. Block		
...		

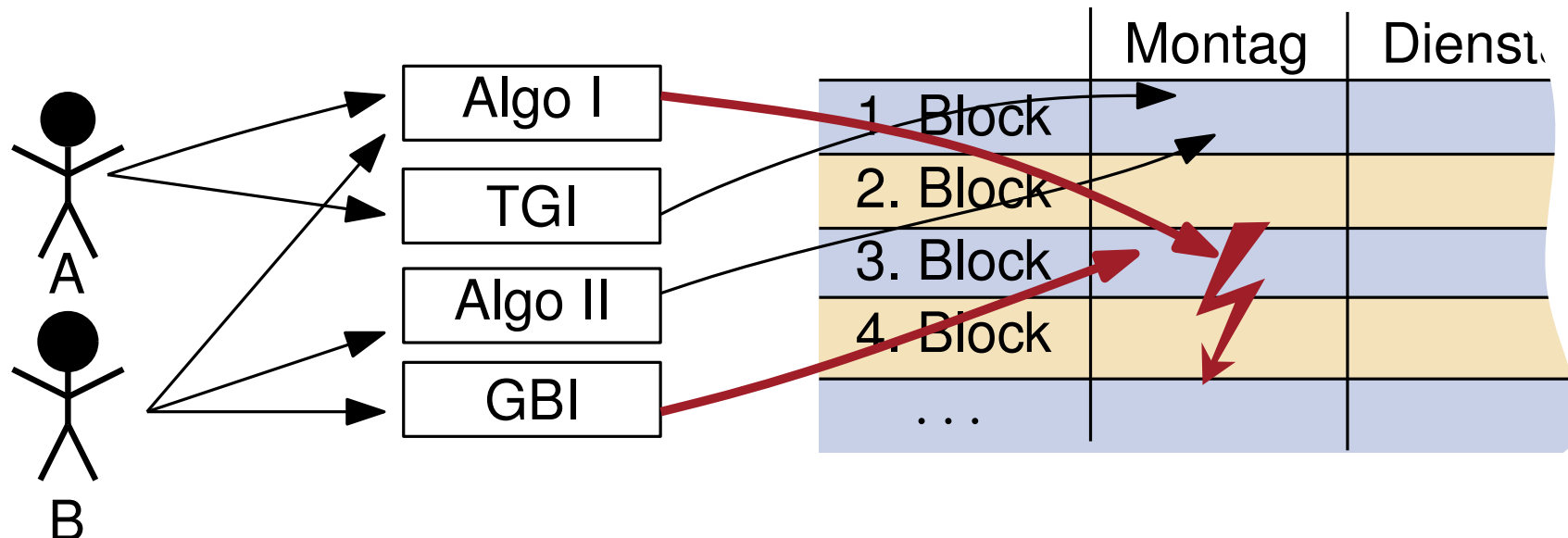
# Klausurplan

## 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:



## 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?

## 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 NP liegt.



## 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 NP liegt.

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

## 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?

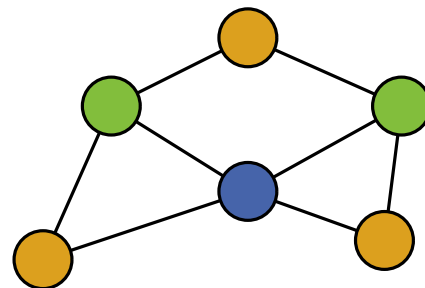
**2. Schritt:** Zeige, dass KLAUSURPLAN NP-schwer ist.

3COLOR  $\propto$  KLAUSURPLAN

## Problem 3COLOR:

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

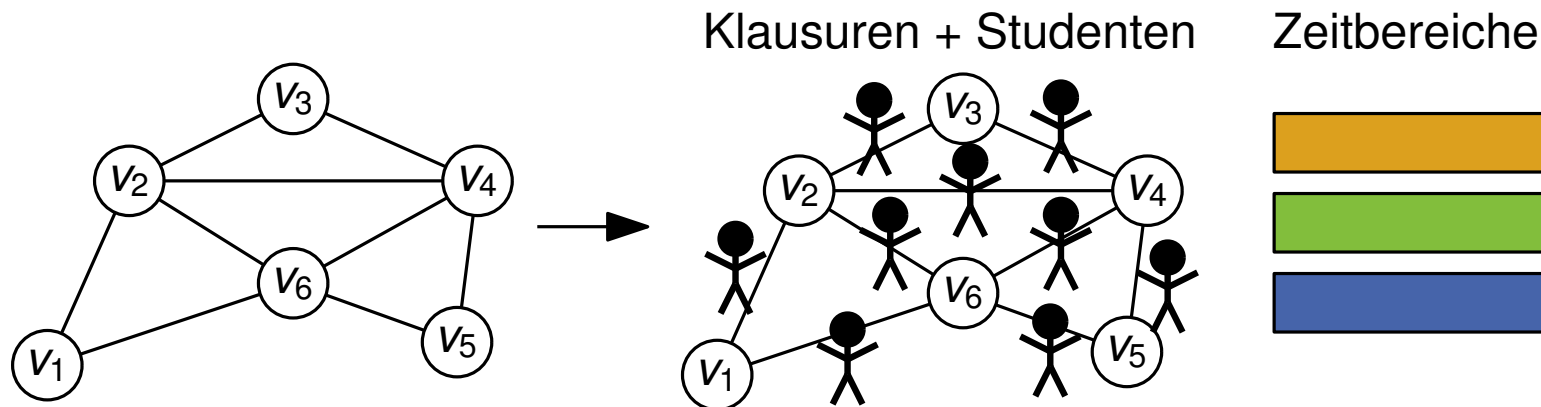
**Frage:** Gibt es eine Knotenfärbung von  $G$  mit höchstens  $k$  Farben, so dass je zwei adjazente Knoten verschiedene Farben besitzen?



**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?

**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

**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.

**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.

**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.

**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.



**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

**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.

**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.

**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.

┆ Zuordnung Klausuren zu Zeitbereichen  
┆ liefert Färbung der Knoten.

**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.

┆ Zuordnung Klausuren zu Zeitbereichen  
┆ liefert Färbung der Knoten.

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

**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.

┆ Zuordnung Klausuren zu Zeitbereichen  
┆ liefert Färbung der Knoten.

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

Jeder Student entspricht einer Kante in  $G$ .

**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.

┆ Zuordnung Klausuren zu Zeitbereichen  
┆ liefert Färbung der Knoten.

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

Jeder Student entspricht einer Kante in  $G$ .

→ Benachbarte Knoten in  $G$  haben unterschiedliche Färbungen.

## Beschränkung eines Problems

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

## Lokales Ersetzen

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

## Komponenten-Design

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



$NP = coNP?$

NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt  $NP = coNP$ .

# NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt NP = coNP.

## Wiederholung:

Die Klasse NP ist die Menge aller Sprachen  $L$ , für die es eine nichtdeterministische Turing-Maschine gibt, die  $L$  in polynomieller Zeit erkennt.

Die Klasse coNP ist die Menge aller Sprachen, deren Komplement in NP enthalten ist:

$$\text{coNP} = \{L \subseteq \Sigma^* \mid \Sigma^* \setminus L \in \text{NP}\}$$

# NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt  $NP = coNP$ .

Sei  $\Pi_1$  ein NP-vollständiges Problem und sei  $\Pi_1^c$  das Komplement hierzu.

# NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt  $NP = coNP$ .

Sei  $\Pi_1$  ein NP-vollständiges Problem und sei  $\Pi_1^c$  das Komplement hierzu.

**Annahme:**  $\Pi_1^c \in NP$

# NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt  $NP = coNP$ .

Sei  $\Pi_1$  ein NP-vollständiges Problem und sei  $\Pi_1^c$  das Komplement hierzu.

**Annahme:**  $\Pi_1^c \in NP$

**Zeige:** Komplement  $\Pi_2^c$  eines beliebigen Problems  $\Pi_2 \in NP$  liegt in NP.

# NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt  $NP = coNP$ .

Sei  $\Pi_1$  ein NP-vollständiges Problem und sei  $\Pi_1^c$  das Komplement hierzu.

**Annahme:**  $\Pi_1^c \in NP$

**Zeige:** Komplement  $\Pi_2^c$  eines beliebigen Problems  $\Pi_2 \in NP$  liegt in NP.

$\Pi_1$  ist NP-vollständig: Es gibt Reduktion  $\varphi$  von  $\Pi_2$  auf  $\Pi_1$   
 $\varphi$  ist auch Reduktion von  $\Pi_2^c$  auf  $\Pi_1^c$

# NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt  $NP = coNP$ .

Sei  $\Pi_1$  ein NP-vollständiges Problem und sei  $\Pi_1^c$  das Komplement hierzu.

**Annahme:**  $\Pi_1^c \in NP$

**Zeige:** Komplement  $\Pi_2^c$  eines beliebigen Problems  $\Pi_2 \in NP$  liegt in NP.

$\Pi_1$  ist NP-vollständig: Es gibt Reduktion  $\varphi$  von  $\Pi_2$  auf  $\Pi_1$   
 $\varphi$  ist auch Reduktion von  $\Pi_2^c$  auf  $\Pi_1^c$

Sei  $I_2^c$  eine Instanz von  $\Pi_2^c$ .

Transformiere die Instanz in eine Instanz von  $\Pi_1^c$ :  $I_1^c = \varphi(I_2^c)$ .



# NP = coNP?

Zeigen Sie: Falls das Komplement eines NP-vollständigen Problems in NP liegt, dann gilt  $NP = coNP$ .

Sei  $\Pi_1$  ein NP-vollständiges Problem und sei  $\Pi_1^c$  das Komplement hierzu.

**Annahme:**  $\Pi_1^c \in NP$

**Zeige:** Komplement  $\Pi_2^c$  eines beliebigen Problems  $\Pi_2 \in NP$  liegt in NP.

$\Pi_1$  ist NP-vollständig: Es gibt Reduktion  $\varphi$  von  $\Pi_2$  auf  $\Pi_1$

$\varphi$  ist auch Reduktion von  $\Pi_2^c$  auf  $\Pi_1^c$

Sei  $I_2^c$  eine Instanz von  $\Pi_2^c$ .

Transformiere die Instanz in eine Instanz von  $\Pi_1^c$ :  $I_1^c = \varphi(I_2^c)$ .

Da  $\Pi_1^c$  in 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.