Grundbegriffe der Theoretischen Informatik

Sommersemester 2018 - Thomas Schwentick

Teil D: Komplexitätstheorie

19: NP-vollständige Probleme

Version von: 28. Juni 2018 (14:08)

Ein erstes NP-vollständiges Problem

- In Teil C der Vorlesung haben wir für die Sprache TM-DIAG die Unentscheidbarkeit durch Diagonalisierung bewiesen und dann alle weiteren unentscheidbaren Probleme durch (direkte oder indirekte) Reduktionen von TM-DIAG gewonnen
- Hinsichtlich **NP**-vollständiger Probleme gehen wir ähnlich vor

- Wir zeigen als nächstes, dass das Problem SAT **NP**-vollständig ist ☞ Satz von Cook
- Für weitere Probleme weisen wir die **NP**-Vollständigkeit dann durch (direkte oder indirekte) polynomielle Reduktionen von SAT nach

Inhalt

Satz von Cook (1/14)

Satz 19.1 [Cook 71]

SAT ist **NP**-vollständig

Beweisskizze

- SAT $\in$ **NP**: ✓

- SAT ist **NP**-schwierig:
  - Wir zeigen, dass für jedes $L \in$ **NP** gilt: $L \leqslant_p$ SAT

- Sei L dazu eine beliebige Sprache aus **NP**
- Sei $M = (Q, \Gamma, \delta, q_1)$ eine (1-String)-TM, die L mit polynomieller Zeitschranke n^k nichtdeterministisch entscheidet
- Sei w eine Eingabe für M und n die Länge von w

Beweisskizze (Forts.)

- Wir zeigen, wie aus w eine KNF-Formel φ konstruiert werden kann, so dass gilt:
  M akzeptiert w nichtdeterministisch
  $\iff \varphi$ ist erfüllbar

- Genauer: wir zeigen, dass es zu jeder Zusatzeingabe y, für die w von M akzeptiert wird, eine erfüllende Belegung α für φ gibt, und umgekehrt

- **Wichtige Idee:**
  - Wir betrachten die Berechnung von M bei Eingabe w als eine *Berechnungstabelle*
    - Die Variablen von φ entsprechen den einzelnen Einträgen der Tabelle
    - Jede Variablenbelegung α der Variablen von φ entspricht einer „ausgefüllten" Tabelle
  - φ soll genau dann wahr werden, wenn α eine akzeptierende Berechnung repräsentiert

Satz von Cook (2/14): Beispiel-TM

Beispiel

- Wie betrachten als Beispiel eine TM M, die die Sprache L aller Strings über $\{a, b\}$, die *kein Palindrom* sind, akzeptiert

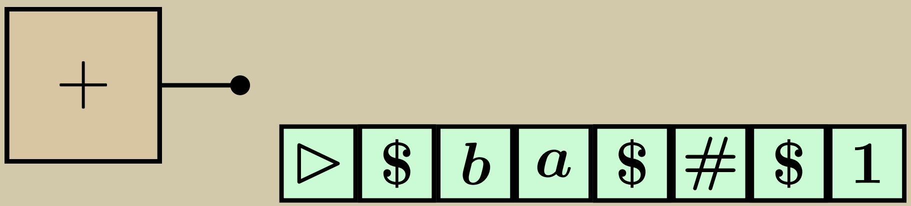
- M erwartet Eingaben der Art $w\#y$
  - w ist der zu überprüfende String ☞ $n \stackrel{\text{def}}{=} |w|$
  - y ist eine Zusatzeingabe der Art 0^k1
- M akzeptiert genau dann, wenn die $(k+1)$-te Position von w von der $(n-k)$-ten Position verschieden ist

Bemerkung

- L ist in **NP**, aber natürlich auch in **P**
- Wir verwenden ein so einfaches Beispiel, weil es sich detailliert auf einer Folie unterbringen lässt

Satz von Cook (3/14): Beispiel-Berechnung

Beispiel

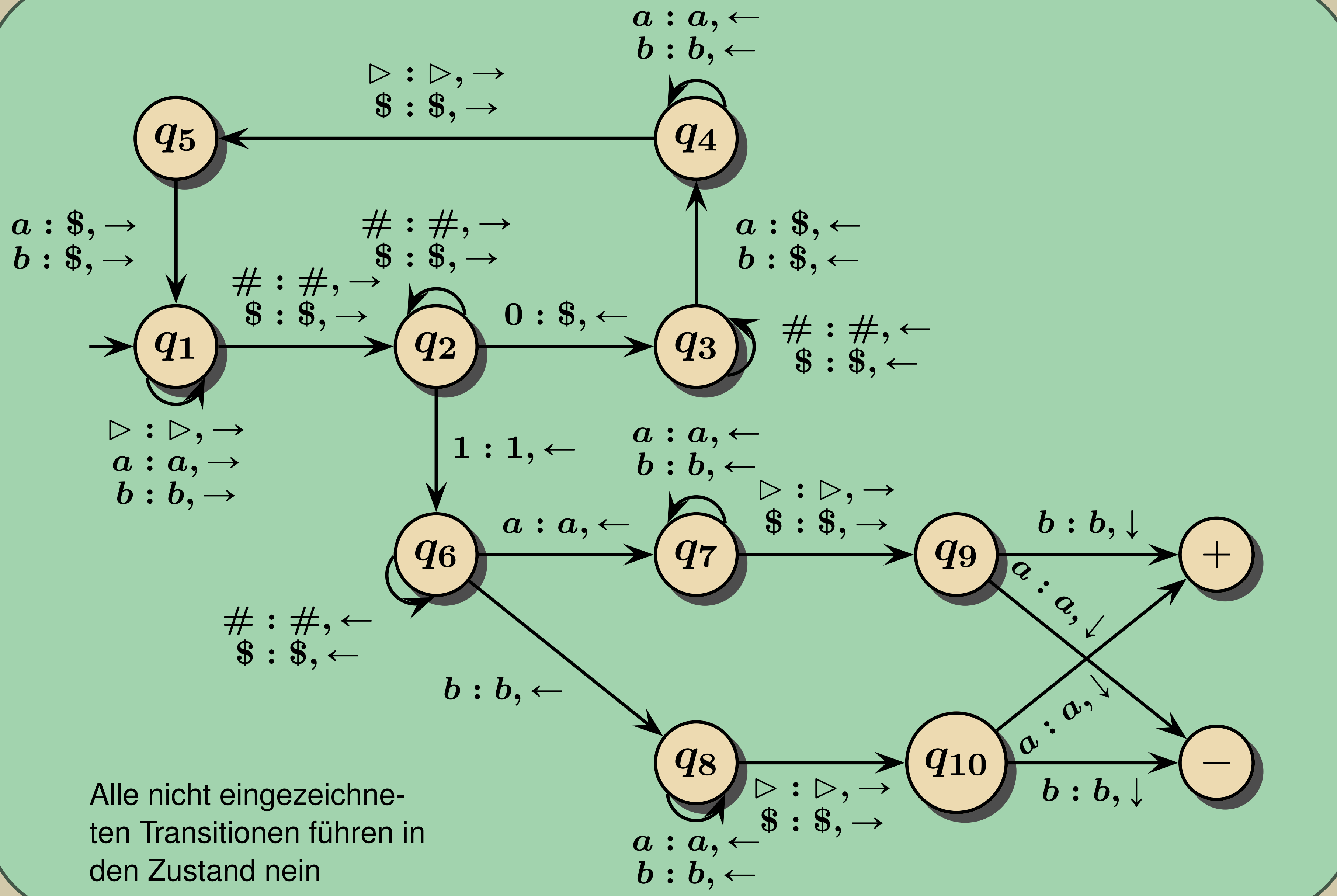


- M erwartet Eingaben der Art $w\#y$

- M bewegt den Kopf zuerst nach rechts auf das erste Zeichen von y
- Falls dort eine 0 steht, wird sie durch $\$$ überschrieben und M läuft nach links und überschreibt dabei das letzte und dann das erste Zeichen von w mit $\$$
- Dann läuft M wieder nach rechts zum nächsten Zeichen von y
- Wenn dort eine 0 steht, werden wieder das letzte und erste noch nicht veränderte Zeichen von w mit $\$$ überschrieben
- Wenn dort eine 1 steht, werden das letzte und erste noch nicht veränderte Zeichen von w verglichen
- Sind diese beiden Zeichen verschieden, so akzeptiert M, andernfalls lehnt M ab

Satz von Cook (4/14): Beispiel-TM als Diagramm

Beispiel



Alle nicht eingezeichneten Transitionen führen in den Zustand nein

Satz von Cook (5/14): Beispiel einer Berechnungstabelle

Beispiel: Tabelle für TM-Beispiel

t	B	Z	P
0	$\triangleright abaa\#01$	q_1	0
1	$\triangleright abaa\#01$	q_1	1
2	$\triangleright abaa\#01$	q_1	2
3	$\triangleright abaa\#01$	q_1	3
4	$\triangleright abaa\#01$	q_1	4
5	$\triangleright abaa\#01$	q_1	5
6	$\triangleright abaa\#01$	q_2	6
7	$\triangleright abaa\#\$1$	q_3	5
8	$\triangleright abaa\#\$1$	q_3	4
9	$\triangleright aba\$\#\$1$	q_4	3
10	$\triangleright aba\$\#\$1$	q_4	2
11	$\triangleright aba\$\#\$1$	q_4	1
12	$\triangleright aba\$\#\$1$	q_4	0
13	$\triangleright aba\$\#\$1$	q_5	1

Beispiel (Forts.)

t	B	Z	P
14	$\triangleright \$ba\$\#\$1$	q_1	2
15	$\triangleright \$ba\$\#\$1$	q_1	3
16	$\triangleright \$ba\$\#\$1$	q_1	4
17	$\triangleright \$ba\$\#\$1$	q_2	5
18	$\triangleright \$ba\$\#\$1$	q_2	6
19	$\triangleright \$ba\$\#\$1$	q_2	7
20	$\triangleright \$ba\$\#\$1$	q_6	6
21	$\triangleright \$ba\$\#\$1$	q_6	5
22	$\triangleright \$ba\$\#\$1$	q_6	4
23	$\triangleright \$ba\$\#\$1$	q_6	3
24	$\triangleright \$ba\$\#\$1$	q_7	2
25	$\triangleright \$ba\$\#\$1$	q_7	1
26	$\triangleright \$ba\$\#\$1$	q_9	2
27	$\triangleright \$ba\$\#\$1$	ja	2

Satz von Cook (6/14): Positionen in der Berechnungstabelle

Beweisskizze (Forts.)

- Zur Vereinfachung nehmen wir oBdA an, dass die Zusatzeingabe genau die Länge $n^k - n - 1$ hat und nur die Zeichen 0 und 1 verwendet
- Es gilt also:
  - $w\#y$ hat genau n^k Zeichen
  - Linker Rand: Position 0
  - w: Positionen $1, \ldots, n$
  - $\#$ an Position $n + 1$
  - y: Positionen $n + 2, \ldots, n^k$

- Klar: in n^k Schritten kann sich der Kopf der Turing-Maschine nicht über y hinaus bewegen

- Sei $\Gamma = \{\sigma_1, \ldots, \sigma_l\}$ das Arbeitsalphabet von M
- Sei $Q = \{q_1, \ldots, q_m\}$ die Zustandsmenge von M inklusive ja, nein und h

Satz von Cook (7/14): Variablen von φ

Beweisskizze (Forts.)

- Die Formel φ verwendet die folgenden aussagenlogischen Variablen:

Variablen	Indizes	Intendierte Bedeutung
$Z_{t,q}$	$t = 0, \ldots, n^k$ $q \in Q$	$\alpha(Z_{t,q}) = 1 \iff$ nach t Schritten befindet sich M im Zustand q
$P_{t,i}$	$t = 0, \ldots, n^k$ $i = 0, \ldots, n^k$	$\alpha(P_{t,i}) = 1 \iff$ nach t Schritten befindet sich der Kopf von M auf Position i
$B_{t,i,\sigma}$	$t = 0, \ldots, n^k$ $i = 0, \ldots, n^k$ $\sigma \in \Gamma$	$\alpha(B_{t,i,\sigma}) = 1$ $\iff$ nach t Schritten befindet sich auf Position i das Zeichen σ

Satz von Cook (8/14): Belegung der Variablen

Beispiel

- Wir werfen einen Blick auf die Variablenbelegung α, die der Berechnungstabelle der Beispielberechnung entspricht
- Wir betrachten nur die Variablen, die die Konfiguration der TM nach 10 Schritten repräsentieren:
  - $t = 10$, $Z = q_4$, $P = 2$
  - $B = \triangleright aba\$\#\$1$

Bemerkung

✎ Im Gegensatz zur Konstruktion im Beweis ist die Länge der Eingabe im Beispiel nicht gleich der Anzahl der Berechnungsschritte
  - Deshalb werden hier, der allgemeinen Semantik von TMs entsprechend, hinter der Eingabe Blanks ergänzt

Beispiel (Forts.)

- $\alpha(Z_{10,q_4}) = 1$
- $\alpha(Z_{10,p}) = 0$, für $p \neq q_4$

- $\alpha(P_{10,2}) = 1$
- $\alpha(P_{10,i}) = 0$, für $i \neq 2$

- $\alpha(B_{10,0,\triangleright}) = 1$
- $\alpha(B_{10,1,a}) = 1$
- $\alpha(B_{10,2,b}) = 1$
- $\alpha(B_{10,3,a}) = 1$
- $\alpha(B_{10,4,\$}) = 1$
- $\alpha(B_{10,5,\#}) = 1$
- $\alpha(B_{10,6,\$}) = 1$
- $\alpha(B_{10,7,1}) = 1$
- $\alpha(B_{10,i,\sqcup}) = 1$, für alle $i > 7$
- $\alpha(B_{10,i,\sigma}) = 0$, für alle übrigen i, σ

Satz von Cook (9/14): Konsistenzbedingungen

Beweisskizze (Forts.)

- φ ist aus mehreren Teilformeln zusammengesetzt:

$$\varphi = \varphi_{\boldsymbol{K}} \wedge \varphi_{\boldsymbol{A}} \wedge \varphi_{\boldsymbol{D}} \wedge \varphi_{\boldsymbol{E}}$$

- $\varphi_{\boldsymbol{K}}$ soll sicherstellen, dass $\boldsymbol{\alpha}$ überhaupt eine Tabelle repräsentiert, also jeder Eintrag der Tabelle genau einmal vorhanden ist

  ☞ Konsistenzbedingungen

- $\varphi_{\boldsymbol{A}}$ drückt aus, dass die erste Zeile der Tabelle der Startkonfiguration entspricht

  ☞ Anfangsbedingungen

- $\varphi_{\boldsymbol{D}}$ drückt aus, dass aufeinander folgende Zeilen der Tabelle mit der Transitionsfunktion verträglich sind

  ☞ Transitionsbedingungen

- $\varphi_{\boldsymbol{E}}$ drückt aus, dass die Berechnung akzeptiert

  ☞ Endbedingung

Satz von Cook (10/14): Konsistenzbedingungen

Beweisskizze (Forts.)

- φ_K drückt für die von α kodierte Tabelle aus:
  - Zu jedem Zeitpunkt ist der Zustand von M eindeutig
  - Zu jedem Zeitpunkt ist die Position des Kopfes von M eindeutig bestimmt
  - Zu jedem Zeitpunkt steht an jeder Stringposition genau ein Zeichen

Beweisskizze (Forts.)

- Wir verwenden dabei die Hilfsformel
$$\psi_{\text{unique}}(x_1, \ldots, x_s) \stackrel{\text{def}}{=} (\bigvee_{i=1}^{s} x_i) \wedge (\bigwedge_{i \neq j} (\neg x_i \vee \neg x_j)),$$
die wahr wird, wenn $\alpha(x_i) = 1$, für genau ein i

- $$\varphi_K \stackrel{\text{def}}{=} \bigwedge_t \psi_{\text{unique}}(Z_{t,q_1}, \ldots, Z_{t,q_m}) \wedge \bigwedge_t \psi_{\text{unique}}(P_{t,0}, \ldots, P_{t,n^k}) \wedge \bigwedge_{t,i} \psi_{\text{unique}}(B_{t,i,\sigma_1}, \ldots, B_{t,i,\sigma_l})$$

Satz von Cook (11/14): Anfangsbedingungen

Beweisskizze (Forts.)

- φ_A beschreibt die Situation von M zum Zeitpunkt 0:
- $\varphi_A \stackrel{\text{def}}{=}$

$$Z_{0,q_1} \wedge P_{0,0} \wedge B_{0,0,\triangleright} \wedge \bigwedge_{i=1}^{n} B_{0,i,w[i]}$$

$$\wedge B_{0,n+1,\#} \wedge \bigwedge_{i=n+2}^{n^k} (B_{0,i,0} \vee B_{0,i,1})$$

- Zu beachten:
  - Die Formel drückt unter anderem aus, dass die Zusatzeingabe nur aus Nullen und Einsen besteht
  - Die Formel legt die Eingabe w fest
    * Dies ist die einzige Teilformel von φ, die wirklich von w abhängt
    * Die anderen Teilformeln hängen allenfalls von der Länge n von w ab
  - Die Formel legt *nicht* die Zusatzeingabe y fest

Satz von Cook (12/14): Transitionsbedingungen

Beweisskizze (Forts.)

- φ_D beschreibt die Beziehung zwischen den aufeinander folgenden Konfigurationen

- $\varphi_D \stackrel{\text{def}}{=} \varphi_{D_1} \wedge \varphi_{D_2}$, wobei:
  - φ_{D_1} beschreibt, was sich an der Stelle, an der sich der Kopf der Turing-Maschine befindet, ändert

- $\varphi_{D_1} \stackrel{\text{def}}{=}$
$$\bigwedge_{t,i,p,\sigma} [(Z_{t,p} \wedge P_{t,i} \wedge B_{t,i,\sigma}) \rightarrow (Z_{t+1,q} \wedge P_{t+1,i+d} \wedge B_{t+1,i,\tau})]$$
- Dabei sind q, τ, d jeweils durch $\delta(p, \sigma) = (q, \tau, d)$ gegeben
  - d wird hier als Zahl in $\{-1, 0, 1\}$ interpretiert ($\leftarrow \equiv -1, \downarrow \equiv 0, \rightarrow \equiv 1$)

✎ φ_{D_1} ist die einzige Teilformel von φ, die von der Transitionsfunktion δ abhängt

Beispiel

- Die Formel φ_{D_1} ist zu lang, um sie für das Beispiel ganz anzugeben
- Deshalb hier nur ein kleiner Ausschnitt für $t = 8, i = 3, p = q_7$
- Es gilt in der TM
  - $\delta(q_7, a) = (q_7, a, \leftarrow)$
  - $\delta(q_7, b) = (q_7, b, \leftarrow)$
  - $\delta(q_7, \$) = (q_9, \$, \rightarrow)$

- Die entsprechende Teilformel lautet dann:
$$[(Z_{8,q_7} \wedge P_{8,3} \wedge B_{8,3,a}) \rightarrow (Z_{9,q_7} \wedge P_{9,2} \wedge B_{9,3,a})] \wedge$$
$$[(Z_{8,q_7} \wedge P_{8,3} \wedge B_{8,3,b}) \rightarrow (Z_{9,q_7} \wedge P_{9,2} \wedge B_{9,3,b})] \wedge$$
$$[(Z_{8,q_7} \wedge P_{8,3} \wedge B_{8,3,\$}) \rightarrow (Z_{9,q_9} \wedge P_{9,4} \wedge B_{9,3,\$})]$$

Satz von Cook (13/14): Transitionsbedingungen (Forts.)

Beweisskizze (Forts.)

- Die Teilformeln
  $$[(Z_{t,p} \wedge P_{t,i} \wedge B_{t,i,\sigma}) \rightarrow (Z_{t+1,q} \wedge P_{t+1,i+d} \wedge B_{t+1,i,\tau})]$$
  von φ_{D_1} sind noch nicht in KNF, lassen sich aber äquivalent umformen:
  $$(\neg Z_{t,p} \vee \neg P_{t,i} \vee \neg B_{t,i,\sigma} \vee Z_{t+1,q}) \wedge$$
  $$(\neg Z_{t,p} \vee \neg P_{t,i} \vee \neg B_{t,i,\sigma} \vee P_{t+1,i+d}) \wedge$$
  $$(\neg Z_{t,p} \vee \neg P_{t,i} \vee \neg B_{t,i,\sigma} \vee B_{t+1,i,\tau})$$

- Ein technisches Detail: Was ist, wenn M weniger als n^k Schritte macht?
  - Dann wird in der Berechnungstabelle die Endkonfiguration ab der entsprechenden Zeile immer wiederholt
  - Die Formel wird analog gebildet (als wäre $\delta(\text{ja}, \sigma) = (\text{ja}, \sigma, \downarrow)$)

Beweisskizze (Forts.)

- φ_{D_2} drückt aus, dass sich der String an allen übrigen Positionen nicht verändert:

- $$\varphi_{D_2} \stackrel{\text{def}}{=} \bigwedge_{t,i,\sigma} ((\neg P_{t,i} \wedge B_{t,i,\sigma}) \rightarrow B_{t+1,i,\sigma})$$
- In konjunktiver Normalform:
  $$\varphi_{D_2} \stackrel{\text{def}}{=} \bigwedge_{t,i,\sigma} (P_{t,i} \vee \neg B_{t,i,\sigma} \vee B_{t+1,i,\sigma})$$

Beispiel

- Für $t = 8$ und $i = 3$ ergibt sich also:
  $$(P_{8,3} \vee \neg B_{8,3,a} \vee B_{9,3,a}) \wedge$$
  $$(P_{8,3} \vee \neg B_{8,3,b} \vee B_{9,3,b}) \wedge$$
  $$\cdots$$
  $$(P_{8,3} \vee \neg B_{8,3,\$} \vee B_{9,3,\$}) \wedge$$
  $$(P_{8,3} \vee \neg B_{8,3,\triangleright} \vee B_{9,3,\triangleright})$$

Satz von Cook (14/14): Abschluss des Beweises

Beweisskizze (Forts.)

- Endbedingung:
  - φ_E drückt aus, dass die letzte Konfiguration den ja-Zustand hat:

$$\varphi_E = Z_{n^k,\text{ja}}$$

- *Behauptung: Die Größe von φ ist polynomiell in n:*
  - Größe der einzelnen Teilformeln:

φ_K	$\mathcal{O}(n^{3k})$
φ_A	$\mathcal{O}(n^{k})$
φ_D	$\mathcal{O}(n^{2k})$
φ_E	$\mathcal{O}(1)$

✎ $|\psi_{\text{unique}}(x_1,\ldots,x_s)| = \mathcal{O}(s^2)$

  - Zu beachten: m und l sind durch M bestimmt und damit konstant

➡ die Größe von φ ist polynomiell in n

- Und: alle Teilformeln sind in konjunktiver Normalform, also auch ihre Konjunktion

Beweisskizze (Forts.)

- Noch zu zeigen: $w \in L \iff \varphi$ erfüllbar
- Also: Zu w gibt es genau dann eine Zusatzeingabe y, die M zum Akzeptieren bringt, wenn φ erfüllbar ist

- Falls es ein solches y gibt, können alle Variablen gemäß ihrer intendierten Bedeutung mit Wahrheitswerten belegt werden

➡ φ wird wahr

- Umgekehrt: Zu jeder erfüllenden Belegung α von φ lässt sich eine Zusatzeingabe y und eine akzeptierende Berechnung von M bei Eingabe w und Zusatzeingabe y konstruieren

➡ $w \in L$

- Nachweis jeweils durch Induktion nach t

- Damit ist die Beweisskizze des Satzes von Cook vollendet

Inhalt

Einleitung

- Wir wissen jetzt also: SAT ist **NP**-vollständig
- Ihre große Bedeutung hat die **NP**-Vollständigkeit erst durch den Nachweis erlangt, dass viele andere algorithmische Probleme **NP**-vollständig sind
- Die erste größere Menge solcher Probleme wurde von Karp 1972 vorgestellt
  - Die Arbeit enthält alle im Folgenden betrachteten Probleme
  - Die hier vorgestellten Beweise sind aber zum Teil anders

- Wie schon gesagt, werden wir ähnlich wie im Falle der unentscheidbaren Probleme in Teil C vorzugehen:
  - Ausgehend von SAT zeigen wir die **NP**-Vollständigkeit der anderen Probleme jeweils mit Hilfe einer *einzelnen* polynomiellen Reduktion

- Zunächst vergewissern wir uns aber, dass dieser Ansatz wirklich funktioniert
- Dazu zeigen wir, dass wir wie folgt schließen können
  - Wenn alle **NP**-Probleme polynomiell reduzierbar auf eine Sprache L' sind
  - und L' polynomiell auf eine Sprache L reduzierbar ist
  - dann sind auch alle **NP**-Probleme polynomiell reduzierbar auf L

- Wir müssen also zeigen, dass $\leqslant_p$ eine transitive Relation ist

Reduktionen und NP-Vollständigkeit (1/2)

Lemma 19.2

- Seien $L_1, L_2, L_3 \subseteq \Sigma^*$ Sprachen
- Falls $L_1 \leqslant_p L_2$ und $L_2 \leqslant_p L_3$, so gilt auch $L_1 \leqslant_p L_3$

Beweisskizze

- Sei f_1 eine polynomielle Reduktion von L_1 auf L_2 und f_2 eine polynomielle Reduktion von L_2 auf L_3

- Behauptung: Die durch $f(w) \stackrel{\text{def}}{=} f_2(f_1(w))$ definierte Funktion ist eine polynomielle Reduktion von L_1 auf L_3

Beweisskizze (Forts.)

- f ist eine Reduktion:
  - Für alle Strings $w \in \Sigma^*$ gilt:
    $$\begin{aligned} & w \in L_1 \\ \iff \quad & f_1(w) \in L_2 \\ \iff \quad & f(w) = f_2(f_1(w)) \in L_3 \end{aligned}$$

- f kann in polynomieller Zeit berechnet werden:
  - Seien n^i und n^j Zeitschranken für die Berechnung von f_1 und f_2
  - ➡ Dann kann $f(w)$ in
    $$\begin{aligned} |w|^i + |f_1(w)|^j &\leqslant |w|^i + (|w|^i)^j \\ &= \mathcal{O}(|w|^{ij}) \end{aligned}$$
    Schritten berechnet werden

- Die binäre Relation $\leqslant_p$ zwischen Sprachen ist also transitiv

Reductionen und NP-Vollständigkeit (2/2)

- Die Möglichkeit des Nachweises der **NP**-Vollständigkeit durch eine einzelne Reduktion wird nun durch das folgende Lemma eröffnet

Lemma 19.3

- Ist L' **NP**-schwierig und gilt $L' \leqslant_p L$, so ist auch L **NP**-schwierig

Beweisskizze

- Sei $L'' \in$ **NP** beliebig

➨ $L'' \leqslant_p L'$

☞ da L' **NP**-schwierig

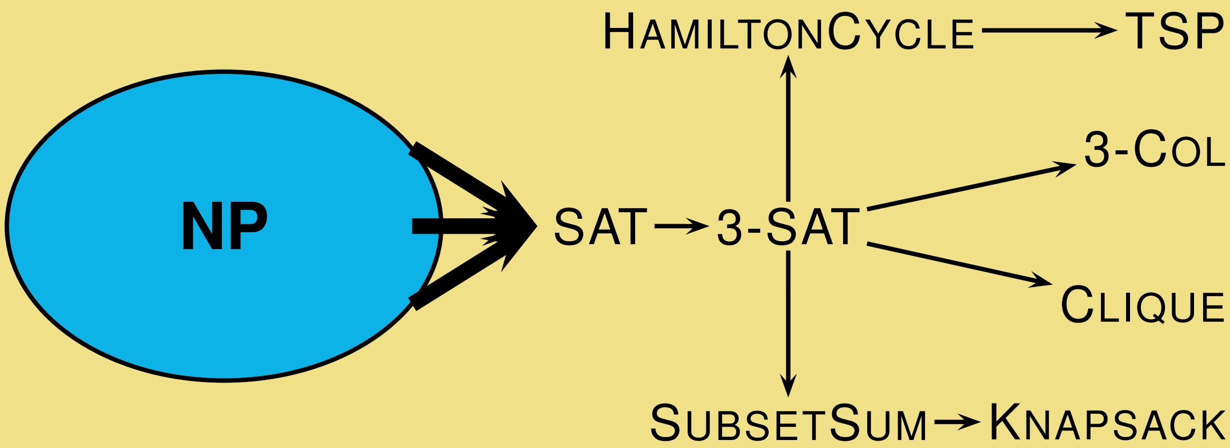
- Wegen $L' \leqslant_p L$ folgt $L'' \leqslant_p L$

☞ gemäß Lemma 19.2

➨ L **NP**-schwierig

- Um nachzuweisen, dass ein Problem L **NP**-vollständig ist, genügt es also zu zeigen:
  - $L \in$ **NP**
  - $L' \leqslant_p L$ für ein **NP**-vollständiges Problem L'

- Die Reduktionen, die wir in diesem Kapitel betrachten werden, sind in der folgenden Abbildung zusammengefasst:



Polynomielle Reduktionen: Rezept

- Die folgenden Beweise für Aussagen der Art $L_1 \leqslant_p L_2$ verlaufen alle nach demselben Muster
- Sie gehen in vier Schritten vor

(1) Definiere eine Reduktionsfunktion f, die Eingaben für L_1 auf Eingaben für L_2 abbildet

(2) Zeige, dass f in polynomieller Zeit berechnet werden kann

✎ Das ist meistens ziemlich offensichtlich

(3) Zeige: wenn $w \in L_1$ dann ist auch $f(w) \in L_2$

- Beweise dazu, dass aus jeder Lösung y_1 für w eine Lösung y_2 für $f(w)$ konstruiert werden kann

(4) Zeige: wenn $f(w) \in L_2$ dann ist auch $w \in L_1$

- Beweise dazu, dass aus jeder Lösung y_2 für $f(w)$ eine Lösung y_1 für w konstruiert werden kann

- Daraus folgt dann $L_1 \leqslant_p L_2$

Inhalt

SAT $\leqslant_p$ 3-SAT (1/4)

Proposition 19.4

$$\text{SAT} \leqslant_p \text{3-SAT}$$

- Bei dieser Reduktion verwenden wir für lange Klauseln eine ähnliche Idee wie beim Entfernen langer rechter Seiten bei der Umwandlung kontextfreier Grammatiken in Chomsky-Normalform...

Beispiel

- Für $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_4 \vee x_7 \vee \neg x_8 \vee x_1 \vee \neg x_5)$ sei:

$$f(\varphi) \stackrel{\text{def}}{=} (x_1 \vee \neg x_2 \vee \neg x_2) \wedge (x_2 \vee \neg x_4 \vee y_1^2) \wedge (\neg y_1^2 \vee x_7 \vee y_2^2) \wedge (\neg y_2^2 \vee \neg x_8 \vee y_3^2) \wedge (\neg y_3^2 \vee x_1 \vee \neg x_5)$$

- Die erfüllende Belegung
  $\theta : x_1 \mapsto 0, x_2 \mapsto 0, x_4 \mapsto 1, x_7 \mapsto 0, x_8 \mapsto 0, x_5 \mapsto 1$
  kann zu einer erfüllenden Belegung von $f(\varphi)$ erweitert werden durch:
  $y_1^2 \mapsto 1, y_2^2 \mapsto 1, y_3^2 \mapsto 0$

SAT $\leqslant_p$ 3-SAT (2/4)

Beweisskizze

- Sei $\varphi = K_1 \wedge \cdots \wedge K_m$ eine KNF-Formel

(1) $f(\varphi) \stackrel{\text{def}}{=} \chi_1 \wedge \cdots \wedge \chi_m$, wobei die 3-KNF-Formeln χ_i wie folgt definiert sind

- Sei $K_i = L_1 \vee \cdots \vee L_j$ (mit Literalen L_ℓ)
- Wenn $j = 1$, dann $\chi_i \stackrel{\text{def}}{=} L_1 \vee L_1 \vee L_1$
- Wenn $j = 2$, dann $\chi_i \stackrel{\text{def}}{=} L_1 \vee L_2 \vee L_2$
- Wenn $j = 3$, dann $\chi_i \stackrel{\text{def}}{=} K_i$
- Wenn $j > 3$ verwenden wir $j - 3$ neue Variablen $y_1^i, \ldots, y_{j-3}^i$ und definieren

$$\begin{aligned}\chi_i \stackrel{\text{def}}{=} (L_1 \vee L_2 \vee y_1^i) \wedge \\ (\neg y_1^i \vee L_3 \vee y_2^i) \wedge \\ \vdots \\ (\neg y_{j-4}^i \vee L_{j-2} \vee y_{j-3}^i) \wedge \\ (\neg y_{j-3}^i \vee L_{j-1} \vee L_j)\end{aligned}$$

(2) $f(\varphi)$ kann in quadratischer Zeit in $|\varphi|$ berechnet werden

SAT $\leqslant_p$ 3-SAT **(3/4)**

Beweisskizze (Forts.)

(3) φ erfüllbar $\Rightarrow f(\varphi)$ erfüllbar:

- Sei θ eine Belegung mit $\theta \models \varphi$

➡ für jedes $i \leqslant m$ gilt: $\theta \models K_i$

- Wir zeigen, dass wir θ zu einer Belegung θ' erweitern können (die auch für die neuen Variablen definiert ist), so dass, für jedes i gilt: $\theta' \models \chi_i$
- Da die neuen Variablen jeweils nur in *einer* Teilformel χ_i vorkommen, können wir die Erweiterung von θ' für jedes χ_i einzeln definieren
- Sei also $i \leqslant m$ und
$$K_i = L_1 \vee \cdots \vee L_j$$
- Falls $j \leqslant 3$, muss θ für χ_i nicht erweitert werden

Beweisskizze (Forts.)

- Sei nun $j > 3$
- Wenn θ eines der beiden ersten Literale von K_i wahr macht ($\theta \models L_1$ oder $\theta \models L_2$), können alle neuen Variablen mit 0 belegt werden:
  - $\theta'(y_\ell^i) \stackrel{\text{def}}{=} 0$, für alle $\ell \leqslant j-3$
- Wenn nicht, aber θ eines der beiden letzten Literale von K_i wahr macht ($\theta \models L_{j-1}$ oder $\theta \models L_j$), können alle neuen Variablen auf 1 gesetzt werden
  - $\theta'(y_\ell^i) \stackrel{\text{def}}{=} 1$, für alle $\ell \leqslant j-3$
- Andernfalls sei $p \leqslant j$ die kleinste Zahl mit $\theta \models L_p$ und wir setzen
  - $\theta'(y_\ell^i) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{für alle } \ell \leqslant p-2 \\ 0 & \text{für alle } \ell \geqslant p-1 \end{cases}$
- In allen Fällen folgt dann: $\theta' \models \chi_i$

SAT $\leqslant_p$ 3-SAT (4/4)

Beweisskizze (Forts.)

(4) $f(\varphi)$ erfüllbar $\Rightarrow \varphi$ erfüllbar:

- Sei θ' eine erfüllende Belegung für $f(\varphi)$

➡ für jedes $i \leqslant m$ gilt: $\theta' \models \chi_i$

- Wir definieren $\theta(x_p) \stackrel{\text{def}}{=} \theta'(x_p)$, für alle Variablen x_p, die in φ vorkommen

- Zu zeigen: für jedes $i \leqslant m$ gilt: $\theta \models K_i$

- Sei also $K_i = L_1 \vee \cdots \vee L_j$ eine Klausel von φ

- Falls $j \leqslant 3$, ist K_i äquivalent zu χ_i und deshalb folgt $\theta \models K_i$ direkt aus $\theta' \models \chi_i$

Beweisskizze (Forts.)

- Beobachtung: falls $j > 3$ können nicht alle Klauseln von χ_i durch Literale mit Variablen y_q^i wahr gemacht werden
  - denn: nach Konstruktion kann jede Variable nur eine Klausel wahr machen, es sind aber weniger neue Variablen als Klauseln

➡ für ein $p \leqslant j$ muss gelten $\theta' \models L_p$

➡ $\theta \models K_i$

Folgerung 19.5

3-SAT ist **NP**-vollständig

Inhalt

3-SAT $\leqslant_p$ CLIQUE (1/5)

- Die Reduktionen 3-COL $\leqslant_p$ SAT und SAT $\leqslant_p$ 3-SAT waren nicht allzu kompliziert
  - Dass sich die Korrektheit einer 3-Färbung eines Graphen in einer aussagenlogischen Formel „kodieren“ lässt, ist nicht allzu überraschend

- Wir wollen jetzt zeigen: 3-SAT $\leqslant_p$ CLIQUE
  - Das ist schon weniger nahe liegend
  - Wie sollen Variablen, Wahrheitsbelegungen und Klauseln in einen Graphen kodiert werden?
  - Das ist deutlich komplizierter

- Grob gesagt, ist die Korrespondenz zwischen Formeln und Graphen in dieser Reduktion wie folgt:
  - Jedes *Vorkommen* eines Literals entspricht einem Knoten im Graphen
  - Kanten zwischen Knoten drücken aus, dass die entsprechenden Literale sich nicht widersprechen
  - Cliquen im Graphen entsprechen dann Mengen simultan erfüllbarer Literale

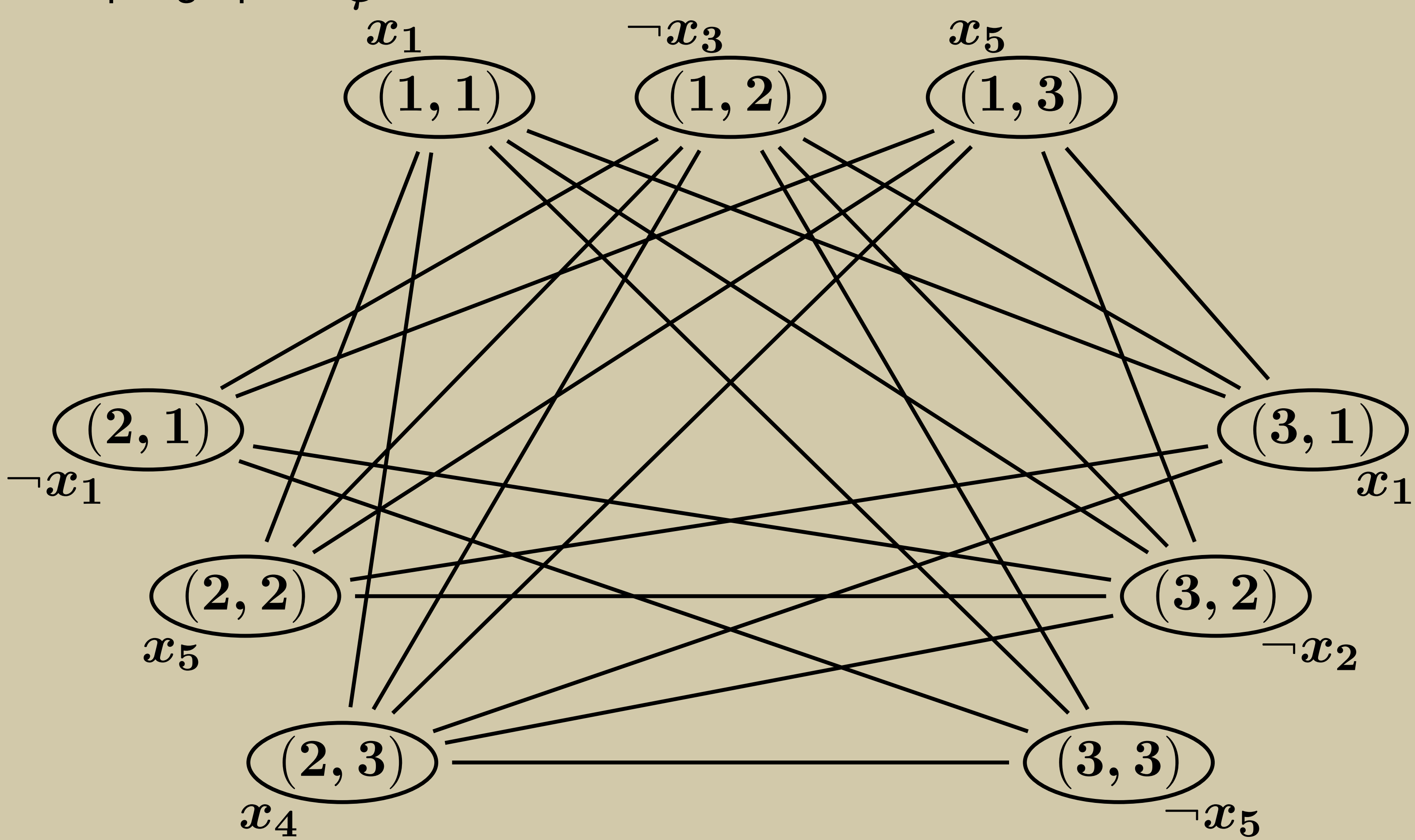
Proposition 19.6

3-SAT $\leqslant_p$ CLIQUE

3-SAT $\leqslant_p$ Clique (2/5)

Illustration der Reduktion von 3-SAT auf Clique

- Beispielformel $\varphi = (x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_5 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_5)$
- Beispielgraph G_φ:



3-SAT $\leqslant_p$ CLIQUE (3/5)

Beweisskizze

(1) Sei φ eine Formel in KNF mit m Klauseln zu je 3 Literalen

- Wir konstruieren einen Graphen G mit $3m$ Knoten so dass gilt:

  G hat eine Clique der Größe $m \iff \varphi$ ist erfüllbar

- Sei $\varphi = (L_{11} \vee L_{12} \vee L_{13}) \wedge \cdots \wedge (L_{m1} \vee L_{m2} \vee L_{m3})$

- Sei $G = (V, E)$ mit
  - $V = \{(i,j) \mid 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant 3\}$
  - $E = \{((i,j),(k,l)) \mid i \neq k$ und $L_{ij} \neq \neg L_{kl}\}$

- G hat also einen Knoten für jedes Vorkommen eines Literals in einer Klausel von φ
- Zwei Knoten sind miteinander verbunden, wenn ihre Literale
  - nicht in derselben Klausel sind *und*
  - sich nicht direkt widersprechen, d.h. nicht einer Variablen x_i und ihrer Negation $\neg x_i$ entsprechen
- Wir definieren $f(\varphi) \stackrel{\text{def}}{=} (G, m)$

(2) f kann in quadratischer Zeit in $|\varphi|$ berechnet werden

3-SAT $\leqslant_p$ Clique (4/5)

Beweisskizze (Forts.)

(3) φ erfüllbar $\Rightarrow$ G hat eine m-Clique

- Sei θ eine erfüllende Belegung für φ
- Dann gibt es für jedes $i \leqslant m$ ein $j_i \in \{1, 2, 3\}$, so dass $\theta \models L_{ij_i}$

➡ Die Literale $L_{1j_1}, \ldots, L_{mj_m}$ sind in verschiedenen Klauseln und widersprechen sich nicht

➡ Ihre zugehörigen Knoten bilden eine Clique der Größe m in G

➡ (3)

Beweisskizze (Forts.)

(4) G hat eine m-Clique $\Rightarrow$ φ erfüllbar

- Sei C eine m-Clique von G
- C besteht aus Knoten zu m Literalen $L_{1j_1}, \ldots, L_{mj_m}$ aus verschiedenen Klauseln
- Da alle diese Literale miteinander verbunden sind, gibt es keine Variable x_p, für die sowohl x_p als auch $\neg x_p$ in $\{L_{1j_1}, \ldots, L_{mj_m}\}$ vorkommt

➡ θ kann so definiert werden, dass alle Literale $L_{1j_1}, \ldots, L_{mj_m}$ wahr werden

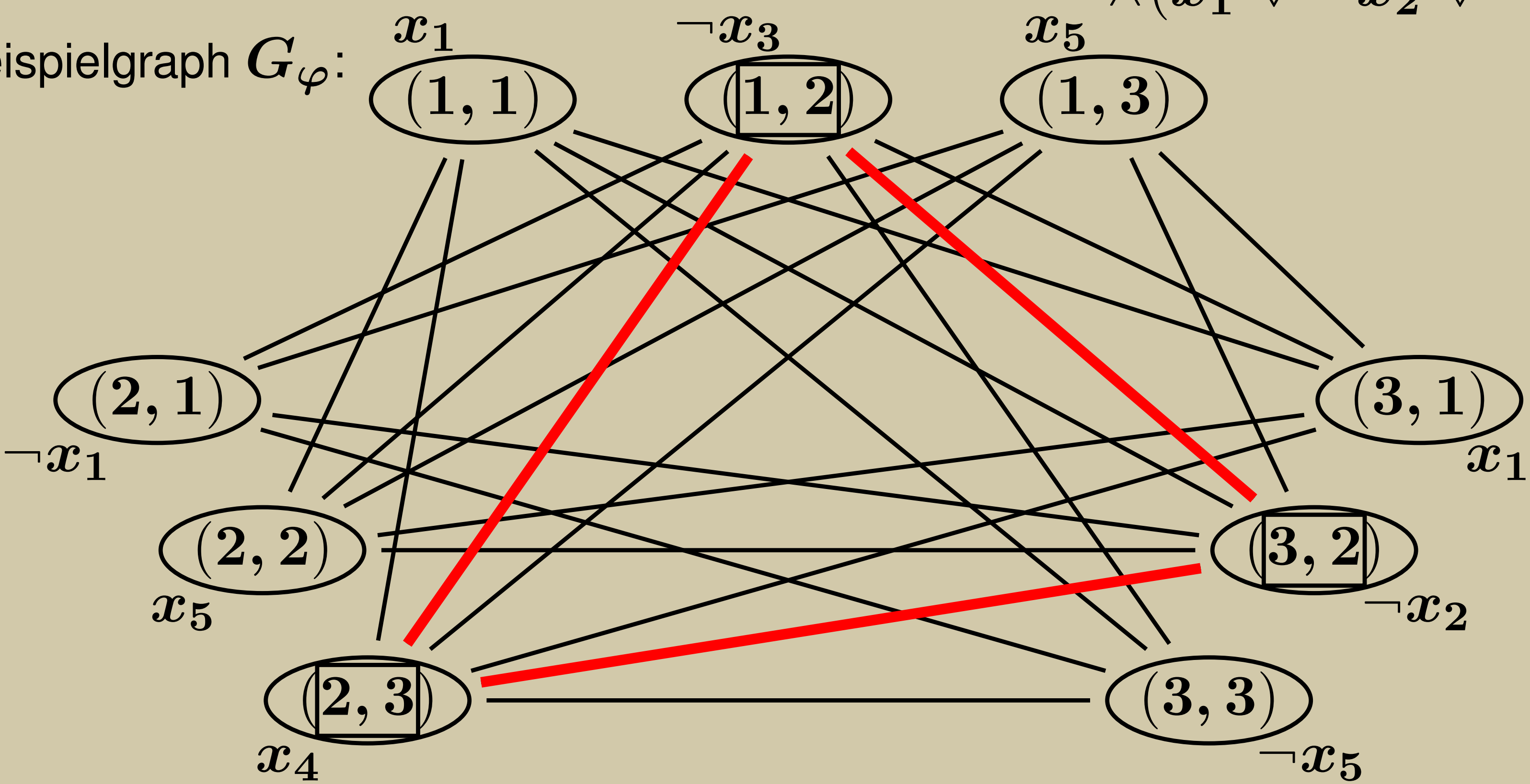
➡ θ macht in jeder Klausel mindestens ein Literal wahr

➡ (4)

3-SAT $\leqslant_p$ Clique (5/5)

Illustration der Reduktion von 3-SAT auf Clique

- Beispielformel $\varphi = (x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_5 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_5)$
- Beispielgraph G_φ:



- Die Knoten $(1,2)$, $(2,3)$ und $(3,2)$ bilden eine Clique in G_φ

➡ Die Literale $\neg x_3$, $\neg x_2$ und x_4 widersprechen sich nicht

➡ Sie induzieren deshalb eine erfüllende Belegung von φ:
  - $\theta(x_2) = 0, \theta(x_3) = 0, \theta(x_4) = 1$
  - Der Rest ist frei wählbar, z.B.: $\theta(x_1) = 0, \theta(x_5) = 1$

Inhalt

Inhalt

Inhalt

Gesamtergebnis

- Da wir für alle in diesem Kapitel betrachteten Probleme schon gezeigt haben, dass sie in **NP** sind, folgt aus den in diesem Kapitel gezeigten Resultaten der folgende Satz

19.12

- Die folgenden Probleme sind **NP**-vollständig:
  - 3-SAT
  - 3-COL
  - CLIQUE
  - HAMILTONCYCLE
  - TSP
  - SUBSETSUM
  - KNAPSACK

Zusammenfassung

- Es gibt Tausende von **NP**-vollständigen Problemen
- Das erste **NP**-vollständige Problem, SAT, haben wir durch den Satz von Cook gewonnen

- **NP**-Vollständigkeitsbeweise verwenden zumeist polynomielle Reduktionen **von** anderen, schon als **NP**-vollständig bekannten, Problemen
- Solche Reduktionsbeweise lassen sich in einer kanonischen Struktur darstellen

Literaturhinweise

Satz von Cook: Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971

- Enthält viele weitere „klassische“ **NP**-vollständige Probleme

Andere NP-vollständige Probleme R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computa- tions*. Plenum, New York, 1972

Gerade Pfade: A. S. Lapaugh and C. H. Papadimitriou. The even-path problem for graphs and digraphs. *Networks*, 14:507–513, 1984

- Die Arbeit zeigt, dass es ein **NP**-vollständiges Problem ist, zu überprüfen, ob es einen Weg gerader Länge zwischen zwei Knoten eines gerichteten Graphen gibt