Komplexitätstheorie

SoSe 2018

Thomas Schneider

Kapitel 3: P versus NP

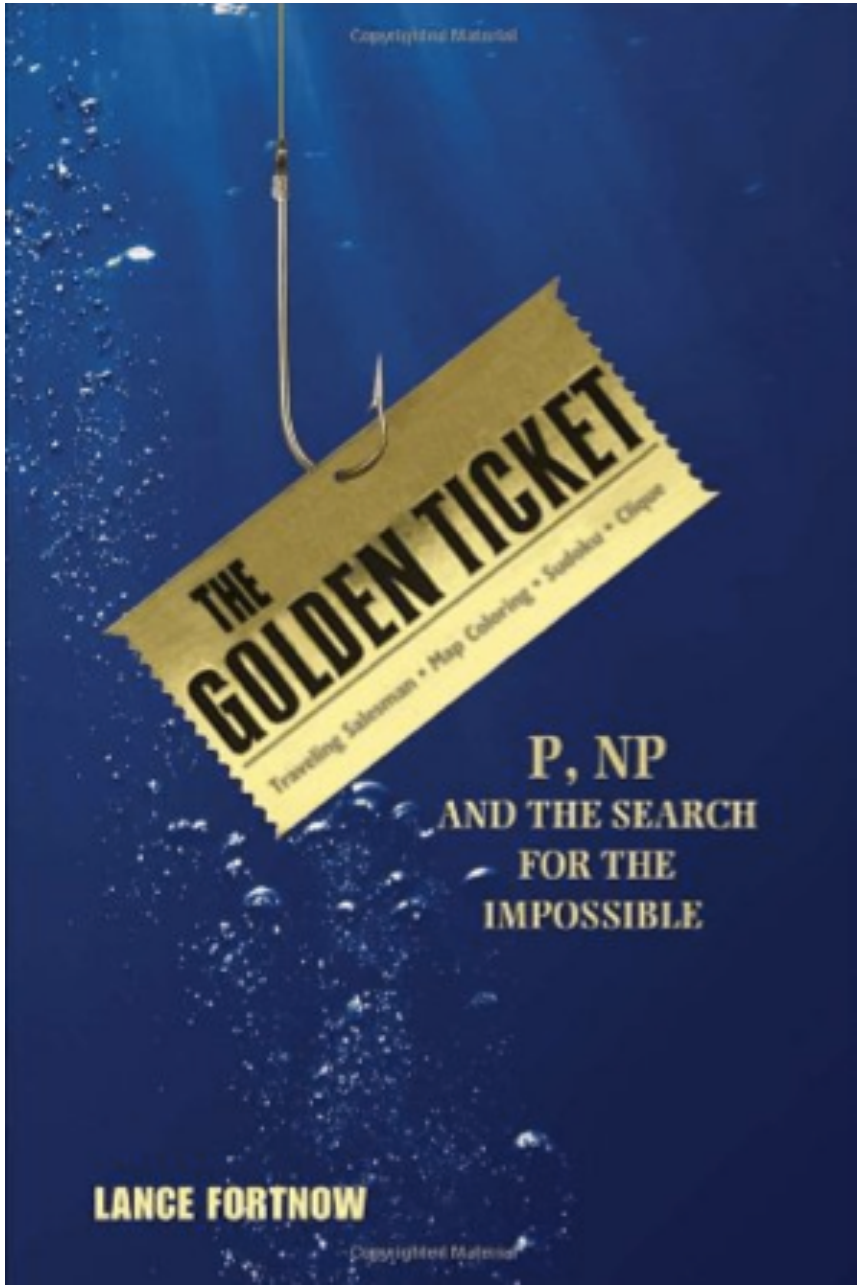
Homepage der Vorlesung: http://tinyurl.com/ss18-kt

P versus NP

Wir definieren die wichtigen Komplexitätsklassen P und NP und studieren deren Zusammenhang:

- Definitionen der Klassen, polynomielle Beweissysteme
- Zusammenhang zwischen NP und Nichtdeterminismus
- NP-Vollständigkeit
- Beispiele für NP-vollständige Probleme
- Komplemente und co-NP
- Isomorphie und spärliche Mengen
- Satz von Ladner und Constraint-Satisfaction-Probleme

Einleitung


THE GOLDEN TICKET
Traveling Salesman • Map Coloring • Sudoku • Clique
P, NP
AND THE SEARCH
FOR THE
IMPOSSIBLE
LANCE FORTNOW

Kapitel 3

Die Klasse P

Traditionelle Sicht:

Problem ist „effizient lösbar“ gdw. es in polynomieller Zeit lösbar ist
(Polynom von beliebigem Grad!)

Entsprechende Komplexitätsklasse:

$$\mathrm{P} := \bigcup_{i \geq 1} \mathrm{DTime}(n^i) \qquad \text{(oft auch PTIME)}$$

Probleme in P nennt man oft auch tractable.

Die Klasse P

Warum ist P eine „gute“ Komplexitätsklasse?

- Zeitkomplexität n^k mit sehr großem k ist zwar bedenklich, aber Polynome mit großem Grad sind fast nie notwendig.
- Nach erweiterter Church-Turing-These ist die Klasse P für alle Berechnungsmodelle identisch (z. B. 1- vs. k-Band-TMs)
- P ist abgeschlossen unter den üblichen Kompositionsoperationen auf Algorithmen. **T3.1**

Die letzten beiden Eigenschaften werden z. B. von DTime(n) nicht erfüllt.

Die Klasse NP

Sehr viele Probleme in der Informatik haben folgende Charakteristik:

Für alle $w \in \Sigma^*$ gilt:

- wenn $w \in L$ (w ist „Ja-Instanz“),

  dann gibt es einen einfach zu verifizierenden Beweis dafür,
  der von kurzer Länge ist (polynomiell in der Länge von w)

- wenn $w \notin L$ (w ist „Nein-Instanz“),

  dann gibt es keinen solchen Beweis.

Diese Beobachtung ist die Grundlage für die Definition der Komplexitätsklasse NP.

Beispiel 1: Cliquenproblem

Beim Cliquenproblem ist der „Beweis“ die Clique selbst:

- wenn $(G, k) \in$ CLIQUE, dann gibt es Knotenmenge $\{v_1, \ldots, v_k\}$, die Clique in G ist;
- wenn $(G, k) \notin$ CLIQUE, dann gibt es keine solche Menge.

Der Beweis ist einfach zu verifizieren: man kann offensichtlich in polynomieller Zeit entscheiden, ob gegebene Knotenmenge Clique ist.

Der Beweis ist kurz: nicht größer als der Graph.

Beispiel 2: Rucksackproblem

Definition Rucksackproblem

Sei $M = \{a_1, \ldots, a_k\}$ eine Menge von *Gegenständen*,
wobei Gegenstand a_i Gewicht g_i und Nutzen n_i hat.

Sei $G \geq 0$ eine Gewichtsgrenze und N ein intendierter Nutzen.

Lösung für Rucksackproblem (M, G, N) ist Teilmenge $R \subseteq M$, so dass

1. $\sum_{a_i \in R} g_i \leq G$ und
2. $\sum_{a_i \in R} n_i \geq N$.

RP ist die Menge aller Instanzen (M, G, N), für die eine Lösung existiert.

T3.2

Beweis für Ja-Instanz (M, G, N) ist Teilmenge $R \subseteq M$, die 1.+2. erfüllt

Offenbar nicht länger als Eingabe & leicht zu verifizieren!

Beispiel 3: Integer Programming

Definition Integer Programming

Ein *lineares Gleichungssystem* G ist eine Menge von Gleichungen der Form

$$c_1 \cdot x_1 + \cdots + c_n \cdot x_n = \alpha,$$

wobei $x_1, \ldots, x_n \in V$, V Variablenmenge, $c_1, \ldots, c_n \in \mathbb{N}$ und $\alpha \in V \cup \mathbb{N}$.

Lösung ist Abbildung $\tau : V \to \mathbb{N}$, so dass alle Gleichungen in G erfüllt sind.

IPROG ist Menge aller Gleichungssysteme G, für die eine Lösung existiert.

T3.3

Gegeben einen Lösungskandidaten kann man offensichtlich in polynomieller Zeit überprüfen, ob er Lösung ist (Addition).

Es ist aber nicht offensichtlich, dass es immer kurze Lösungen/Beweise für IPROG gibt ($\mathbb{N}$ hat Elemente unbeschränkter Größe).

Beispiel 3: Integer Programming

Ohne Beweis:

Theorem (Papadimitriou)

Sei G Gleichungssystem mit m Gleichungen, n Variablen sowie Konstanten, deren Werte durch a beschränkt sind.

Dann gibt es Lösung gdw. es eine Lösung $\tau : V \to \mathbb{N}$ gibt mit $\tau(x) \leq n(ma)^{2m+1}$ für alle $x \in V$.

→ **Also kann man als „kleine Beweise“ verwenden:**

- Lösungen, deren Zahlenwerte wie im o. g. Theorem beschränkt sind
- Zahlenwerte sind exponentiell in der Größe von G, also ist deren binäre Repräsentation nur polynomiell groß ($\log(2^n)$ ist polynomiell!)

Die Klasse NP

Es gibt Tausende solcher Probleme …

Wir setzen:

„einfach zu verifizieren“: in polynomieller Zeit

„kurzer Beweis“: Länge des Beweises polynomiell in Länge der Instanz

Daraus ergibt sich die Definition der Klasse NP.

Die Klasse NP

Definition Komplexitätsklasse NP

Sei $L \subseteq \Sigma^*$. Relation $R \subseteq \Sigma^* \times \Gamma^*$ ist *Beweissystem* für L, wenn

- $(w, b) \in R$ impliziert $w \in L$ und (Korrektheit)
- $w \in L$ impliziert $(w, b) \in R$ für ein $b \in \Gamma^*$ (Vollständigkeit)
  so ein b heißt *Beweis* oder *Zeuge* für w

R ist *polynomiell*, wenn

- es gibt Polynom p, so dass $|b| \leq p(|w|)$ für alle $(w, b) \in R$
- $R \in$ P (d. h.: gegeben $w \in \Sigma^*$ und $b \in \Gamma^*$
  kann DTM in polynomieller Zeit entscheiden, ob $(w, b) \in R$)

NP ist Klasse aller Probleme L, für die es polynomielles Beweissystem gibt.

Die Beispiele haben gezeigt:

CLIQUE, RP, IPROG sind in NP.

Kapitel 3

P versus NP

Theorem

$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{ExpTime}, \quad \text{wobei } \mathrm{ExpTime} := \bigcup_{i \geq 1} \mathrm{DTIME}\left(2^{\mathcal{O}(n^i)}\right).$

T3.4

Sehr unbefriedigend:

Für viele wichtige Probleme in NP (z. B. CLIQUE, RP, IPROG) ist unbekannt, ob sie in P (also effizient lösbar) sind!

Es ist sogar unbekannt, ob P ≠ NP (und NP ≠ ExpTime)

P versus NP

Intuitive Formulierung der P-versus-NP-Frage:

Ist das Finden eines Beweises schwieriger als das Überprüfen?

Unsere Intuition sagt ja, also $P \neq NP$!

In der Tat glaubt eigentlich niemand, das $P = NP$,
aber ein Beweis konnte seit 50 Jahren nicht geführt werden!

Die P-versus-NP-Frage …

- ist eines der wichtigsten offenen Probleme der Informatik
- wurde vom *Clay Mathematics Institute* als eins von sieben Millenium Prize Problems ausgelobt

P versus NP

Konsequenzen eines Beweises von

- P = NP: dramatisch.

  Tausende bisher als sehr schwierig eingestufter Probleme wären dann wohl effizient lösbar
  (z. B. CLIQUE, RP, IPROG)

- P ≠ NP: weniger dramatisch – aber wichtig zu wissen:

  Möglicherweise interessante Konsequenzen in Kryptographie
  (dort: schwierige Probleme nützlich statt ärgerlich!)

Kapitel 3

Die Klasse NP

Die Klasse NP hat viele äquivalente Charakterisierungen:

- polynomielle Beweissysteme
- nichtdeterministische Turingmaschinen
- existentielle Logik zweiter Stufe (Satz von Fagin)
- randomisierte Beweissysteme (PCP-Theorem)
- etc.

Nichtdeterministische TMs

Nichtdeterministische Turingmaschinen (NTMs) generalisieren DTMs:

- Statt Übergangs*funktion* δ haben sie Übergangs*relation*

$$\Delta \subseteq (Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \times Q \times \Gamma \times \{L, R, N\}$$

- (q, a, q', a', L) heißt: wenn M in q ist und a liest, so kann sie nach q' gehen, a' schreiben und sich nach links bewegen
- *Relation*: NTM hat u. U. mehrere Möglichkeiten für nächsten Schritt
- Konfigurationen / Berechnungen definiert wie für DTMs
- Es gibt jetzt mehrere Berechnungen auf derselben Eingabe
- NTM *akzeptiert* Eingabe w,
  wenn mindestens eine Berechnung auf w akzeptierend

Nichtdeterministische TMs

Definition NTime

Für NTM M und $w \in \Sigma^*$ schreiben wir $\text{time}_M(w) = n$,
wenn alle Berechnungen von M auf w höchstens n Schritte umfassen.

Sei $t : \mathbb{N} \to \mathbb{N}$ monoton wachsende Funktion mit $t(n) \geq n$.
M ist *t-zeitbeschränkt*, wenn $\text{time}_M(w) \leq t(n)$ für alle w der Länge n.

Zeitkomplexitätsklasse $\text{NTime}_i(t)$ ist definiert als Menge der Probleme

$$\{L \subseteq \Sigma^* \mid \exists \mathcal{O}(t)\text{-zeitbeschr. } i\text{-Band-NTM } M \text{ mit } L(M) = L\}.$$

Wir setzen $\text{NTime}(t) := \bigcup_{i \geq 1} \text{NTime}_i(t)$.

Theorem (NTM-Charakterisierung von NP)

$$\text{NP} = \bigcup_{i \geq 1} \text{NTime}(n^i)$$

T3.5

Nichtdeterministische TMs

Klassischerweise ist diese Charakterisierung sogar
die eigentliche Definition von NP.

Die P-versus-NP-Frage kann also auch wie folgt verstanden werden:

Kann eine nichtdeterministische Maschine in polynomieller Zeit
mehr erreichen als eine deterministische Maschine?

Kapitel 3

NP-Härte

Bisher ist es für sehr viele „typische“ NP-Probleme nicht gelungen zu zeigen, ob sie in P (effizient lösbar) sind oder nicht.

Der Begriff der NP-Härte liefert ein Mittel, solche Probleme trotzdem als schwierig klassifizieren zu können.

Starke Indikation dafür, dass Problem *L* nicht in P ist:

1. *L* ist „mindestens so schwierig“ wie alle anderen Probleme in NP
2. Daraus folgt, dass *L* nur in P ist, wenn P = NP (also unwahrscheinlich!)

Reduktionen

Reduktionen sind zentral in der Komplexitätstheorie:

- Werkzeug zum Vergleich verschiedener Probleme, nicht nur im Kontext von NP
- existieren in zahllosen Variationen (für verschiedene Zwecke)

Definition (polynomielle) Reduktion

Sei $L \subseteq \Sigma^*$, $L' \subseteq \Gamma^*$. Eine *Reduktion* von L auf L' ist berechenbare Funktion $f : \Sigma^* \to \Gamma^*$, so dass

$$w \in L \text{ gdw. } f(w) \in L' \quad \text{für alle } w \in \Sigma^*.$$

f heißt *polynomiell*, wenn f in polynomieller Zeit berechenbar ist.

Wir schreiben $L \leq_p L'$, wenn polynomielle Reduktion von L auf L' existiert.

T3.6

Reduktionen

Intuitiv: $L \leq_p L'$ bedeutet „L' ist mindestens so schwer wie L“.

Denn: Algorithmus für L' kann mit nur polynomiellem Mehraufwand zum Lösen von L verwendet werden!

Theorem

1. P ist abgeschlossen unter polynomiellen Reduktionen:
   $L' \in \mathsf{P}$ und $L \leq_p L'$ impliziert $L \in \mathsf{P}$.
2. NP ist abgeschlossen unter polynomiellen Reduktionen:
   $L' \in \mathsf{NP}$ und $L \leq_p L'$ impliziert $L \in \mathsf{NP}$.
3. „Polynomiell reduzierbar“ ist transitive Relation:
   $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$ impliziert $L_1 \leq_p L_3$.

NP-Härte

Definition NP-Härte, NP-Vollständigkeit

Problem L ist

- *NP-hart*, wenn $L' \leq_p L$ für alle $L' \in$ NP;
- *NP-vollständig*, wenn L sowohl NP-hart als auch in NP.

Ein NP-hartes Problem ist also mindestens so schwer
wie jedes andere Problem in NP.

Beobachtung

Wenn L NP-hart und $L \in$ P, dann P $=$ NP.

Solange das P-versus-NP-Problem ungelöst ist,
wird NP-Härte deshalb als „nicht effizient lösbar“ gewertet.

Kapitel 3

NP-Härte

Zwei Ansätze, NP-Härte von Problem L zu zeigen:

1. Zeigen, dass $L' \leq_p L$ für alle $L' \in$ NP
2. Zeigen, dass $L' \leq_p L$ für ein NP-hartes Problem L'

Lemma

Wenn L NP-hart ist und $L \leq_p L'$, dann ist L' NP-hart.

T3.7

SAT

Definition Formel, Wertzuweisung, Erfüllbarkeit

Sei VAR unendlich abzählbare Menge von *Aussagenvariablen*.

Menge der *aussagenlogischen Formeln* (kurz: AL-Formeln)
ist die kleinste Menge, so dass:

- Jedes $x \in$ VAR ist AL-Formel.
- Wenn φ, ψ AL-Formeln, so auch $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$.

Wertzuweisung (kurz: WZ) ist Abbildung V : VAR $\to \{0, 1\}$.

WZ V wird wie folgt auf zusammengesetzte AL-Formeln erweitert:

- $V(\neg\varphi) = 1 - V(\varphi)$
- $V(\varphi \wedge \psi) = \min\{V(\varphi), V(\psi)\}$
- $V(\varphi \vee \psi) = \max\{V(\varphi), V(\psi)\}$

AL-Formel φ ist *erfüllbar*, wenn es WZ V gibt mit $V(\varphi) = 1$.

T3.8

Der Satz von Cook

Problem **SAT:** die Menge aller erfüllbaren AL-Formeln

Theorem (Cook/Levin aka „Satz von Cook“ aka „Cook’s Theorem“)

SAT ist NP-vollständig.

SAT $\in$ NP: $R = \{(\varphi, V) \mid V$ ist WZ für Variablen in φ, die φ erfüllt $\}$
ist polynomielles Beweissystem

NP-Härte: Zeigen, dass Wortproblem jeder polyzeit-beschränkten NTM polynomiell auf SAT reduziert werden kann.

(Bekannt aus „Theoretische Informatik 2“)

Der Satz von Cook

Sei $L \in$ NP und M eine $p(n)$-zeitbeschränkte NTM mit $L(M) = L$.

Ziel: gegeben w, finden von AL-Formel φ_w so dass

1. M akzeptiert w gdw. φ_w erfüllbar und
2. φ_w in Zeit polynomiell in $|w|$ konstruiert werden kann

Akzeptierende Berechnung von M auf $w = a_1 \cdots a_n$ ist Matrix:

$\triangleright$	q_0, a_1	a_2	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\triangleright$	b	q, a_2	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\triangleright$	b	q', b'	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$0..p(n)$

$0..p(n)+1$

Der Satz von Cook

Repräsentation der Matrix mittels Variablen:

- $B_{a,i,t}$: zum Zeitpunkt t ist Zelle i mit a beschriftet;
- $K_{i,t}$: zum Zeitpunkt t ist der Kopf über Zelle i;
- $Z_{q,t}$: zum Zeitpunkt t ist q der aktuelle Zustand

für alle $t \leq p(n) + 1$, $i \leq p(n)$, $a \in \Gamma$, $q \in Q$.

Gesuchte Formel φ_w ist Konjunktion folgender Formeln:

Berechnung beginnt mit Startkonfiguration für $w = a_1 \cdots a_n$:

$$\psi_{\text{ini}} := Z_{q_0,0} \wedge K_{1,0} \wedge B_{\triangleright,0,0} \wedge \bigwedge_{1 \leq i \leq n} B_{a_i,i,0} \wedge \bigwedge_{n < i \leq p(n)} B_{\perp,i,0}.$$

Der Satz von Cook

Sei $R(i) = i + 1, \quad N(i) = i, \quad L(i) = i - 1$ wenn $i > 1, \quad L(0) = 0.$

Schritte folgen der Übergangsrelation:

$$\psi_{\mathsf{move}} := \bigwedge_{t<p(n), q\in Q\setminus\{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}, a\in\Gamma, i\leq p(n)+1} \Big((Z_{q,t} \wedge K_{i,t} \wedge B_{a,i,t}) \rightarrow \bigvee_{(q,a,q',a',M)\in\Delta, M(i)\leq p(n)+1} (Z_{q',t+1} \wedge K_{M(i),t+1} \wedge B_{a',i,t+1}) \Big)$$

Zellen ohne Kopf ändern sich nicht:

$$\psi_{\mathsf{keep}} := \bigwedge_{t<p(n), i\leq p(n)+1, a\in\Gamma} \big(\neg K_{i,t} \wedge B_{a,i,t} \rightarrow B_{a,i,t+1} \big)$$

Eingabe wird akzeptiert:

$$\psi_{\mathsf{acc}} := \bigvee_{t\leq p(n)} \big(Z_{q_{\mathsf{acc}},t} \wedge \bigwedge_{t'<t} \neg Z_{q_{\mathsf{rej}},t'} \big)$$

Der Satz von Cook

Bandbeschriftung, Kopfposition, Zustand sind eindeutig und definiert:

$$\psi_{\text{aux}} := \bigwedge_{t,q,q',q\neq q'} \neg(Z_{q,t} \wedge Z_{q',t}) \wedge \bigwedge_{t,i,a,a',a\neq a'} \neg(B_{a,i,t} \wedge B_{a',i,t}) \wedge \bigwedge_{t,i,j,i\neq j} \neg(K_{i,t} \wedge K_{j,t})$$

$$\bigwedge_{t\leq p(n)} \bigvee_{q\in Q} Z_{q,t} \wedge \bigwedge_{t\leq p(n)} \bigvee_{i\leq p(n)+1} K_{i,t} \wedge \bigwedge_{t\leq p(n),i\leq p(n)+1} \bigvee_{a\in\Gamma} B_{a,i,t}$$

Wir setzen nun

$$\varphi_w := \psi_{\text{ini}} \wedge \psi_{\text{move}} \wedge \psi_{\text{keep}} \wedge \psi_{\text{acc}} \wedge \psi_{\text{aux}}.$$

Leicht zu sehen:

φ hat Länge $\mathcal{O}(p(n)^2)$, kann in Zeit $\mathcal{O}(p(n)^2)$ generiert werden.

Lemma

φ_w erfüllbar gdw. M akzeptiert w

Kapitel 3

3SAT

Oft ist es jedoch bequemer, nur Formeln in bestimmter Form zu betrachten.

Definition 3SAT

Eine *3-Formel* hat die Form

$$\bigwedge_i (\ell_{1,i} \vee \ell_{2,i} \vee \ell_{3,i})$$

wobei die $\ell_{j,i}$ *Literale* sind, also Variable oder deren Negation.

Jede Disjunktion $(\ell_{1,i} \vee \ell_{2,i} \vee \ell_{3,i})$ heißt *Klausel*.

3SAT ist die Menge aller erfüllbaren 3-Formeln.

T3.9

Bekannt aus „Theoretische Informatik 2“:

3SAT ist NP-vollständig.

Hier nur kurze Erinnerung.

3SAT

Theorem

3SAT ist NP-vollständig.

3SAT $\in$ NP: klar (Spezialfall von SAT)

Beweis NP-Härte: polynomielle Reduktion von SAT

Gegeben AL-Formel φ, konstruiere in polynomieller Zeit 3-Formel ψ so dass φ erfüllbar gdw. ψ erfüllbar.

Idee:

- führe Aussagenvariable für jede Teilformel von φ ein
- beschreibe das Verhalten von Teilformeln in AL
- Konvertiere entstehende Formel in 3-Formel
- Resultierende Formel ist nicht äquivalent, aber äqui-erfüllbar

T3.10

CLIQUE

Ebenfalls bekannt aus „Theoretische Informatik 2“:

Theorem

CLIQUE ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

3-Färbbarkeit

Wie CLIQUE sind viele interessante Probleme auf Graphen NP-vollständig.

Wir betrachten ein **weiteres Beispiel.**

Definition 3-Färbbarkeit

Ungerichteter Graph $G = (V, E)$ ist *3-färbbar*, wenn es Abbildung

$$f : V \to \{R, G, B\}$$

gibt, so dass $\{v, v'\} \in E$ impliziert $f(v) \neq f(v')$ *(3-Färbung)*.

3F ist die Menge aller Graphen G, die 3-färbbar sind.

T3.11

Leicht zu sehen: 3F $\in$ NP

3-Färbbarkeit

Theorem

3F ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

Gegeben 3-Formel φ, konstruiere in polynomieller Zeit Graph G so dass φ erfüllbar gdw. G 3-färbbar

Idee:

- Führe pro Literal einen Knoten ein.
- Verwende die Farben „wahr“, „falsch“ und „hilf“.
- Verbinde komplementäre Literale mit Kanten, um konsistente WZ zu erzwingen.
- Verwende Teilgraphen, um Erfülltsein jeder Klausel zu erzwingen.

T3.12

NP-vollständige Probleme aus anderen Bereichen

NP-vollständige Probleme gibt es nicht nur in der Logik und Graphentheorie, sondern in vielen weiteren Bereichen der Informatik.

1972 veröffentlichte Richard Karp in einem berühmten Aufsatz 21 solche (und sehr unterschiedliche) Probleme.

Im Buch „Computers and Intractability“ von Garey und Johnson finden sich ca. 300 NP-vollständige Probeme; heute kennt man Tausende!

Wir betrachten **zwei weitere Beispiele:**

- Integer Programming
- Rucksackproblem

Zur Erinnerung:

Definition Integer Programming

Ein *lineares Gleichungssystem* G ist eine Menge von Gleichungen der Form

$$c_1 \cdot x_1 + \cdots + c_n \cdot x_n = \alpha,$$

wobei $x_1, \ldots, x_n \in V$, V Variablenmenge, $c_1, \ldots, c_n \in \mathbb{N}$ und $\alpha \in V \cup \mathbb{N}$.

Lösung ist Abbildung $\tau : V \to \mathbb{N}$, so dass alle Gleichungen in G erfüllt sind.

IPROG ist Menge aller Gleichungssysteme G, für die eine Lösung existiert.

Integer Programming

Theorem

IPROG ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

Gegeben 3-Formel φ, konstruiere in polynomieller Zeit Gleichungssystem G so dass φ erfüllbar gdw. G Lösung hat.

Idee:

- Verwende für jedes Literal eine numerische Variable.
- Stelle durch Gleichungen sicher, dass diese Variablen konsistente Werte aus dem Bereich {0,1} erhalten.
- Stelle durch zusätzliche Gleichungen sicher, dass jede Klausel mindestens ein wahres Literal enthält.

Rucksackproblem (Wdhlg.)

- Es soll ein Rucksack mit gegebener Kapazität gepackt werden.
- Es gibt eine Menge von zu packenden Gegenständen mit unterschiedlichem Wert und Gewicht.
- Man möchte Gegenstände von maximalem Gesamtwert mitnehmen.

Definition Rucksackproblem

Sei $M = \{a_1, \ldots, a_k\}$ eine Menge von *Gegenständen*, wobei Gegenstand a_i Gewicht g_i und Nutzen n_i hat.

Sei $G \geq 0$ eine Gewichtsgrenze und N ein intendierter Nutzen.

Lösung für Rucksackproblem (M, G, N) ist Teilmenge $R \subseteq M$, so dass

1. $\sum_{a_i \in R} g_i \leq G$ und
2. $\sum_{a_i \in R} n_i \geq N$.

RP ist die Menge aller Instanzen (M, G, N), für die eine Lösung existiert.

Rucksackproblem

Theorem

RP ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

Gegeben 3-Formel φ, konstruiere in polynomieller Zeit Rucksackproblem (M, G, N) so dass φ erfüllbar gdw. (M, G, N) Lösung hat.

Idee:

- Verwende für jede Variable x_i zwei Gegenstände a_i und $\overline{a}_i$.
- Wenn x_i wahr ist, dann ist a_i im Rucksack, aber $\overline{a}_i$ nicht; wenn x_i falsch ist, dann umgekehrt.
- Für jeden Gegenstand a_i ist Gewicht g_i gleich Nutzen n_i und $G = N$.
- Zusätzlich werden Gegenstände b_i und c_i für jede Klausel benötigt.

Rucksackproblem

Beweisdetails.

Sei φ 3-Formel mit ℓ Variablen $x_1, \ldots, x_\ell$ und k Klauseln.

Konstruiere Instanz (M, G, N) von RP wie folgt.

- $M = \{a_1, \overline{a}_1, \ldots, a_\ell, \overline{a}_\ell, \;\; b_1, c_1, \ldots, b_k, c_k\}$
- Für alle Gegenstände ist Nutzen $=$ Gewicht; außerdem $N = G$.
- Alle Nutzenwerte sind $(\ell + k)$-stellige Dezimalzahlen wie folgt:

  Nutzen von a_i ($\overline{a}_i$):

  – Dezimalstelle i ist 1; Rest der ersten ℓ Dezimalstellen ist 0.

  – Dezimalstelle $\ell + j$ ist 1, wenn Literal x_i ($\neg x_i$) in j-ter Klausel auftritt, sonst 0.

  Nutzen von b_i, c_i:

  – Dezimalstelle $\ell + i$ ist 1, alle anderen 0.

$G\,(=N)$ hat Dezimaldarstellung $\underbrace{11\cdots1}_{\ell}\underbrace{33\cdots3}_{k}$

T3.13

NP-Vollständigkeit

NP-vollständige Probleme finden sich auch im täglichen Leben:

- **Minesweeper:** Gegeben den momentanen Stand eines Spiels (mit Brett beliebiger Größe), ist an einer bestimmten (verdeckten) Stelle mit Sicherheit eine Mine versteckt?
- **Sudoku:** Gegeben ein partiell gelöstes Sudoku, kann es zu einem vollständig gelösten erweitert werden?
- **Bundesliga:** Gegeben den momentanen Punktestand der Bundesliga (mit beliebig vielen Mannschaften), kann eine bestimmte Mannschaft noch Meister werden?
- …

Kapitel 3

Motivation

Sei $\overline{3F}$ die Menge aller nicht 3-färbbaren Graphen.

In welcher Komplexitätsklasse ist $\overline{3F}$?

- Wenn $\overline{3F}$ in NP ist, müsste es polynomielles Beweissystem geben.

  Aber was ist polynomieller Beweis dafür, dass Graph nicht 3-färbbar?
  Menge aller möglichen 3-Färbungen? Exponentiell viele!

- Offensichtlich ist $\overline{3F}$ ein Problem, bei dem es kurze und einfach zu verifizierende Beweise für nein-Instanzen gibt statt für ja-Instanzen.

Komplemente von Problemen

Definition Komplement (Problem)

Sei $L \subseteq \Sigma^*$ Problem. Das *Komplement* von L ist $\overline{L} := \Sigma^* \setminus L$.

Beispiele:

- $\overline{\text{3F}}$
- $\overline{\text{SAT}}$ (Menge aller unerfüllbaren AL-Formeln)
- $\overline{\text{CLIQUE}}$ (Menge aller Paare (G, k), so dass G keine k-Clique hat)

Deterministische Zeitkomplexitätsklassen wie z. B. P sind unter Komplement abgeschlossen:

Lemma

Sei $\mathcal{C} = \text{DTime}(t)$ für beliebiges t. Dann gilt: $L \in \mathcal{C}$ impliziert $\overline{L} \in \mathcal{C}$.

Also z. B. für GAP (Erreichbarkeitsproblem auf gerichteten Graphen):
$\overline{\text{GAP}} \in \text{P}$

T3.14

Komplement-Komplexitätsklassen

Argument aus dem letzten Beweis schlägt fehl für nichtdeterministische TMs, funktioniert also nicht für die alternative Charakterisierung von NP!

Definition Komplement-Komplexitätsklasse

Sei $\mathcal{C}$ Komplexitätsklasse. $\text{co}\mathcal{C} := \{L \mid \overline{L} \in \mathcal{C}\}$

Wir interessieren uns hier insbesondere für die **Klasse coNP.**

Z. B. sind $\overline{\text{3F}}, \overline{\text{SAT}}, \overline{\text{CLIQUE}}$ per Definition in coNP.

Weiteres natürliches coNP-Problem:

Definition Gültigkeit in Aussagenlogik

AL-Formel φ ist *gültig* gdw. jede WZ φ erfüllt.

Lemma

Gültigkeit ist in coNP.

T3.15

coNP-Vollständigkeit

Härte und Vollständigkeit sind definiert wie für NP:

Definition coNP-Härte, coNP-Vollständigkeit

Ein Problem L ist

- *coNP-hart*, wenn $L' \leq_p L$ für alle $L' \in$ coNP;
- *coNP-vollständig*, wenn L coNP-hart und in coNP.

Lemma

L ist NP-hart gdw. $\overline{L}$ coNP-hart.

$\overline{\text{3F}}, \overline{\text{SAT}}, \overline{\text{CLIQUE}}$ etc. sind also coNP-vollständig.

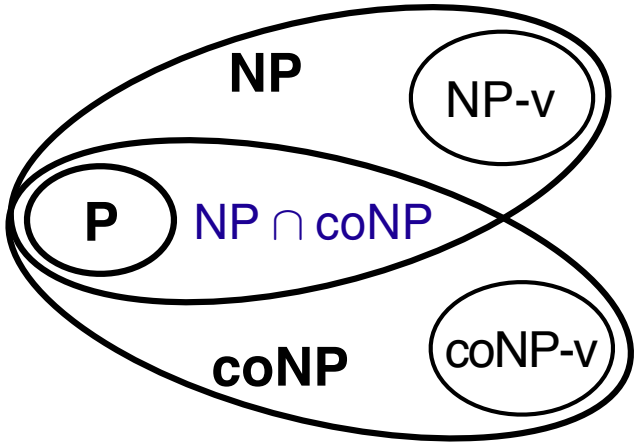
T3.16

Lemma

Gültigkeit ist coNP-vollständig.

T3.17

P versus NP versus coNP



Leicht zu sehen:

$P \subseteq coNP$ **T3.18**

Es wird vermutet:

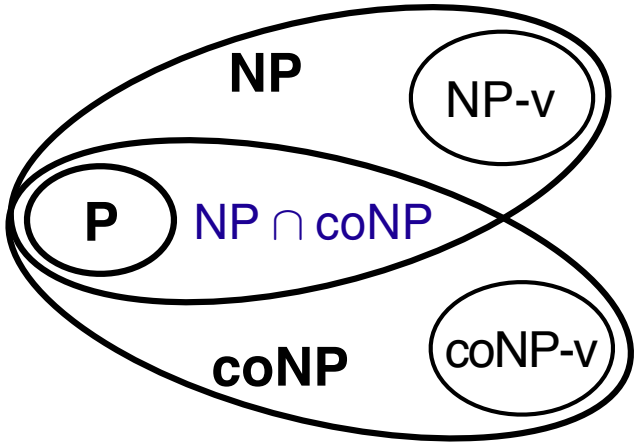
1. $NP \neq coNP$
2. $NP \cap coNP \neq P$

Beide Vermutungen implizieren $P \neq NP$:

Wenn $P = NP$, dann $P = NP = coNP$ (Abschluss P unter Komplement)

Die umgekehrte Implikation ist nicht bekannt.

NP ∩ coNP



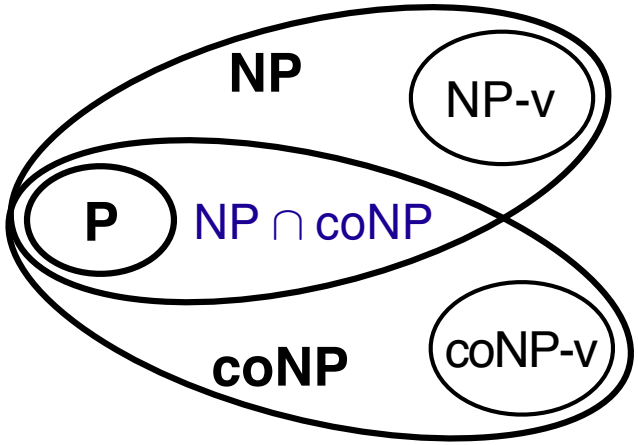
Damit ergibt sich NP ∩ coNP als weitere interessante Klasse.

NP: kurze/einfach zu verifizierende Beweise für ja-Instanzen

coNP: kurze/einfach zu verifizierende Beweise für nein-Instanzen

NP ∩ coNP: beides ist der Fall

NP ∩ coNP



Ein bekanntes Problem in NP ∩ coNP:

Definition Integer-Faktorisierung

Integer-Faktorisierung (IF) ist das folgende Problem:

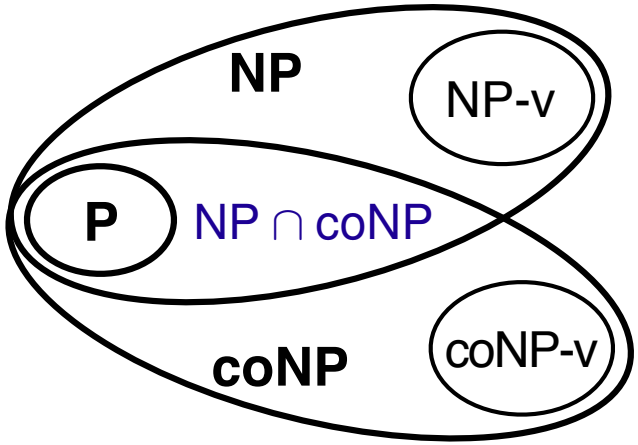
Gegeben $n, k \in \mathbb{N}$, entscheide, ob n einen Faktor $\leq k$ hat.

Theorem

IF $\in$ NP ∩ coNP

T3.19

NP ∩ coNP und NP-Vollständigkeit



Probleme in NP ∩ coNP sind . . .

- wahrscheinlich nicht NP-vollständig:

Lemma

Wenn es ein NP-vollständiges $L \in \text{NP} \cap \text{coNP}$ gibt, dann $\text{NP} = \text{coNP}$.

T3.20

Kryptographie mit dem RSA-Verfahren ist nicht sicher, wenn $\text{IF} \in \text{P}$ (denn mit binärer Suche ließen sich dann effizient Faktoren finden).

- natürliche Kandidaten für NP-intermediate Probleme (siehe §3.9)

Kapitel 3

Isomorphievermutung

NP-vollständige Probleme scheinen alle einander ähnlich zu sein.

Beobachtung

Wenn L, L' NP-vollständig, dann $L \leq_p L'$ und $L' \leq_p L$.

Die Reduktionen sind aber nicht unbedingt strukturerhaltend: **T3.21**

- Injektivität nicht gewährleistet

  z. B. 3SAT $\leq_p$ 3F:
  Verdoppeln von Klauseln in 3-Formel ändert konstruierten Graphen nicht.

- Surjektivität nicht gewährleistet

  z. B. 3SAT $\leq_p$ 3F: nur Graphen mit bestimmter Struktur konstruiert

- Reduktionen $L \leq_p L'$ und $L' \leq_p L$ sind unabhängig voneinander

Wie ähnlich sind NP-vollständige Probleme einander genau?

Isomorphievermutung

Hier

- fordern wir Injektivität und Surjektivität
- verschmelzen wir die zwei unabhängigen Reduktionen zu einer bidirektionalen

Definition p-Isomorphismus

Polynomielle Reduktion $f : \Sigma^* \to \Gamma^*$ von L' auf L ist *p-Isomorphismus*, wenn

- f Bijektion ist (also injektiv und surjektiv);
- nicht nur f, sondern auch f^{-1} in Polyzeit berechenbar ist.

Existiert so ein f, dann sind L' und L *p-isomorph*.

T3.22

Leicht zu sehen: f^{-1} ist Polyzeit-Reduktion von L auf L'

Isomorphievermutung

Vermutung (Berman/Hartmanis 1977)

Alle NP-vollständigen Probleme sind paarweise p-isomorph.

Intuitiv sagt diese Vermutung: alle NP-vollständigen Probleme sind dasselbe Problem, in unterschiedlicher „Verkleidung“.

Sollte die Vermutung wahr sein, so ist sie nicht leicht zu beweisen:

Theorem

Wenn alle NP-vollständigen Probleme paarweise p-isomorph sind, dann $P \neq NP$.

T3.23

Es ist recht unklar, ob die Isomorphievermutung wahr oder falsch ist.

Alle bekannten NP-vollständigen Probleme sind aber p-isomorph.

Der Satz von Mahaney

Im Zusammenhang mit der Isomorphievermutung
steht folgendes Theorem über Probleme mit wenigen ja-Instanzen.

Definition

Eine Menge $L \subseteq \Sigma^*$ heißt *spärlich*, wenn es ein Polynom p gibt, so dass L für jede Wortlänge n höchstens $p(n)$ Wörter der Länge n enthält.

Beachte: wenn $|\Sigma| > 1$, dann erhält Σ^n exponentiell viele Wörter.
Spärliche Mengen sind also Probleme mit sehr wenigen ja-Instanzen.

Satz von Mahaney (1980)

Wenn es eine spärliche Menge gibt, die NP-vollständig ist, dann P = NP.

Man lernt also etwas über die Struktur von NP-vollständigen Problemen.

Der Satz von Mahaney

Für einen Beweis betrachten wir eine geeignete Variante von SAT.

Sei $<$ die lexikographische Ordnung aussagenlogischer WZ. **T3.24**

Definition

LeftSAT ist die Menge aller Paare (φ, V), wobei

- φ eine aussagenlogische Formel ist und
- V eine WZ (für die Variablen in φ),

so dass es eine WZ $V' \leq V$ gibt, die φ erfüllt.

T3.25

Man überlegt sich leicht: LeftSAT ist NP-vollständig.

Beweis Satz von Mahaney:

Wir nehmen an, dass es eine spärliche NP-vollständige Menge S gibt, und zeigen, dass LeftSAT dann in P ist. Also gilt P = NP. **T3.26**

Kapitel 3

NP-intermediate Probleme

Naheliegende Frage:

Ist jedes NP-Problem in P oder NP-vollständig?

Das ist wahrscheinlich nicht der Fall, wie sich bereits bei NP ∩ coNP andeutete.

Definition NP-intermediate

Probleme $L \in \text{NP} \setminus \text{P}$, die nicht NP-vollständig sind, heißen *NP-intermediate*.

NPI ist die Klasse aller NP-intermediate Probleme.

Satz von Ladner (1975)

Wenn $\text{P} \neq \text{NP}$, dann $\text{NPI} \neq \emptyset$.

Also ist NP-Vollständigkeit auch dann wichtig, wenn $\text{P} \neq \text{NP}$ bewiesen ist!

NP-intermediate Probleme

Satz von Ladner (1975)

Wenn P $\neq$ NP, dann NPI $\neq \emptyset$.

Überblick Beweis:

- subtiles Diagonalisierungsargument
- Starte mit SAT; entferne auf systematische Weise ja-Instanzen
  → man erhält „SAT mit Löchern“ (SATmL)
- Erreiche damit, dass jede mögliche Reduktion von SAT auf SATmL fehlschlägt → SATmL nicht NP-hart
- Halte die Löcher so klein wie möglich,
  damit SATmL trotzdem nicht in Polyzeit gelöst werden kann.

(Details: siehe Arora & Barak, §3.3)

NP-intermediate Probleme

Bisher konnte kein natürliches Problem in NPI identifiziert werden.

Kandidaten:

- Graph-Isomorphismus: gegeben ungerichtete Graphen G_1, G_2, kann man die Knoten in G_2 so umbenennen, dass $G_1 = G_2$?
- Log-Clique: enthält ein gegebener Graph mit n Knoten eine Clique der Größe $\log(n)$?
- Alle Probleme in NP $\cap$ coNP, z. B. IF

Deutlich stärkere Variante des Satzes von Ladner:

Satz von Ladner, reloaded

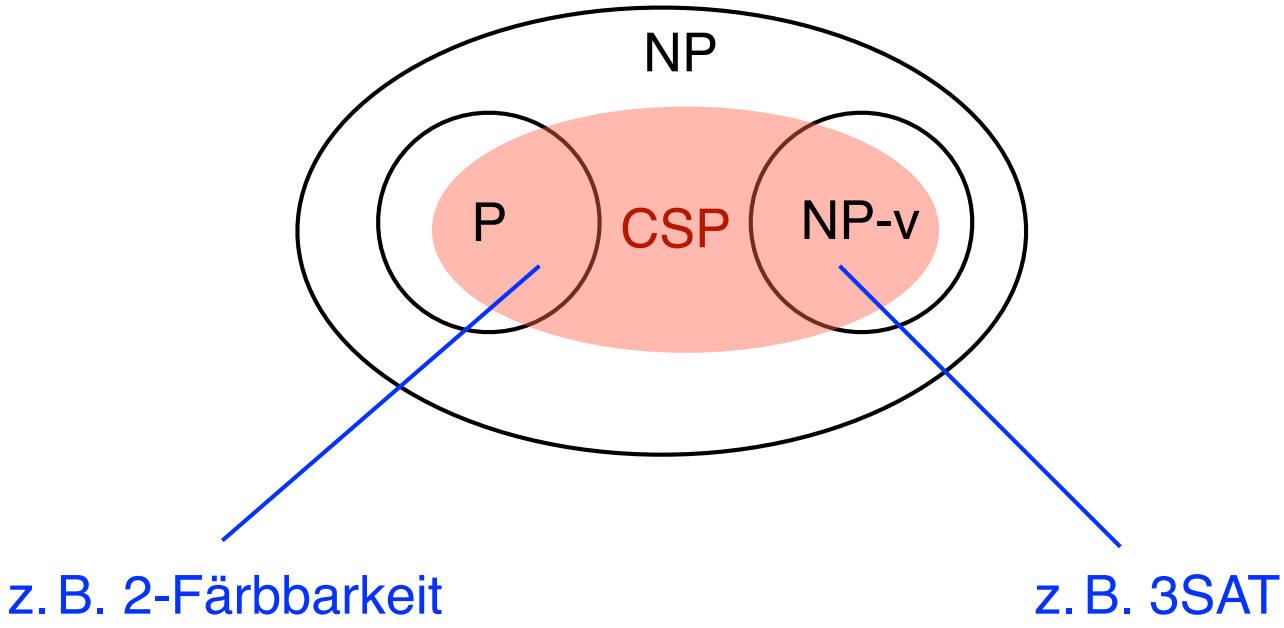
Wenn P $\neq$ NP, dann gibt es unendliche Folge $L_1, L_2, \ldots \in$ NPI, so dass für alle $i \geq 1$ gilt: $L_{i+1} \leq_p L_i$ und $L_i \not\leq_p L_{i+1}$.

T3.27

Constraint Satisfaction

Constraint-Satisfaction-Probleme (CSPs):

Natürliche Teilklasse von NP, in der viele wichtige Probleme enthalten sind



Es besteht Grund zur Annahme, dass CSP leichter zu analysieren ist als NP.

Insbesondere wurde lange vermutet, dass es kein CSP gibt,
das NP-intermediate ist.

Update Okt. 2017: Vermutung durch Bulatov/Zhuk bewiesen (später mehr)

Constraint Satisfaction

CSPs kommen ursprünglich aus der künstlichen Intelligenz.

Beispiel: Lösen von Kryptogrammen:

$$\begin{array}{rr} & S\ E\ N\ D \\ + & M\ O\ R\ E \\ \hline = & M\ O\ N\ E\ Y \end{array}$$

Kann man den Buchstaben Ziffern so zuweisen, dass die Rechnung aufgeht?

Weitere Beispiele:

Konfigurationsprobleme, Bildinterpretation, Scheduling-Probleme usw.

CSPs haben stark kombinatorischen Charakter.

Constraint Satisfaction

Ein CSP ist definiert durch:

- endlichen *Wertebereich* wie z.B. $\{0, 1, \ldots, 9\}$
- die erlaubten *Arten von Constraints*, z.B.

  einstellige Constraints der Form "$x = 0$" (x, x_i Variablen)

  fünfstellige Constraints der Form "$x_1 + x_2 + x_3 = 10x_4 + x_5$"

Eingabe für das CSP besteht aus endlicher Menge von Constraints, z. B.:

$$z = 0$$

$$x_D + x_E + z = 10c_1 + x_Y$$

$$x_N + x_R + c_1 = 10c_2 + x_E$$

usw.

$$\begin{array}{rr} & S\ E\ N\ D \\ + & M\ O\ R\ E \\ \hline = & M\ O\ N\ E\ Y \end{array}$$

und ist ja-Instanz, wenn man den Variablen Werte so zuweisen kann, dass alle Constraints erfüllt sind.

Constraint Satisfaction

Definition Constraint-Satisfaction-Problem

Eine *relationale Struktur* $\mathfrak{A}$ besteht aus

- einem nichtleeren *Wertebereich* A und
- einer Menge $\mathcal{R}$ von *Relationssymbolen* R mit zugeordneter *Stelligkeit* n_R und zugeordneter *Relation* $R^{\mathfrak{A}} \subseteq A^{n_R}$.

Jede endliche relationale Struktur $\mathfrak{A}$ definiert Problem **CSP($\mathfrak{A}$)** wie folgt:

Eingabe: Menge M von *Constraints* der Form $R(x_1, \ldots, x_n)$, wobei $R \in \mathcal{R}$ n-stellig und $x_1, \ldots, x_n$ Variablen

Ausgabe: „ja“, wenn es *Lösung* für M gibt, d. h.

Abbildung $\delta : \mathrm{VAR}(M) \to A$, so dass
für alle $R(x_1, \ldots, x_n) \in M$ gilt: $(\delta(x_1), \ldots, \delta(x_n)) \in R^{\mathfrak{A}}$

und „nein“ sonst

T3.28

Constraint Satisfaction

Sei CSP die Klasse aller Constraint-Satisfaction-Probleme.

Lemma

CSP $\subseteq$ NP.

T3.29

Viele bekannte NP-vollständige Probleme sind als CSPs darstellbar:

3F, k-Färbbarkeit für jedes feste $k > 2$, 3SAT,
IPROG über jedem festen Bereich $\{0, \ldots, n\}$ usw.

T3.30

Es gibt aber auch NP-Probleme, die nicht als CSPs darstellbar sind.

Das sieht man mittels Homomorphismen zwischen relationalen Strukturen.

Homomorphismen

Definition Homomorphismus

Seien $\mathfrak{A}, \mathfrak{B}$ relationale Strukturen über derselben Menge von Relationssymbolen.

Homomorphismus von $\mathfrak{B}$ nach $\mathfrak{A}$ ist Abbildung $h : B \to A$, so dass:

$$\text{Wenn } (b_1, \ldots, b_n) \in R^{\mathfrak{B}}, \quad \text{dann } (h(b_1), \ldots, h(b_n)) \in R^{\mathfrak{A}}.$$

Wir schreiben $\mathfrak{B} \to \mathfrak{A}$, wenn es Homomorphismus von $\mathfrak{B}$ nach $\mathfrak{A}$ gibt.

T3.31

CSPs und Homomorphismen

Auch jede Eingabe für ein CSP($\mathfrak{A}$) (also jede Constraintmenge M) kann als relationale Stuktur $\mathfrak{B}_M$ aufgefasst werden:

- Der Wertebereich von $\mathfrak{B}_M$ besteht aus den Variablen in M.
- Die Relationssymbole und Stelligkeiten sind dieselben wie in $\mathfrak{A}$.
- $R^{\mathfrak{B}_M} = \{(v_1, \ldots, v_n) \mid R(v_1, \ldots, v_n) \in M\}$

Äquivalente Definition Constraint-Satisfaction-Problem

Jede relationale Struktur $\mathfrak{A}$ definiert Problem CSP($\mathfrak{A}$) wie folgt:

$$\text{CSP}(\mathfrak{A}) = \{\mathfrak{B} \mid \mathfrak{B} \to \mathfrak{A}\}$$

T3.32

CSP versus NP

Wir können nun den **Zusammenhang zwischen CSP und NP** untersuchen:

Lemma

Jedes CSP($\mathfrak{A}$) ist abgeschlossen unter homomorphen Urbildern, d. h.:

Wenn $\mathfrak{B}$ ja-Instanz und $\mathfrak{B}' \to \mathfrak{B}$, dann ist $\mathfrak{B}'$ ja-Instanz.

Beweis ist nun einfach, denn
Homomorphismen sind unter Komposition abgeschlossen.

T3.33

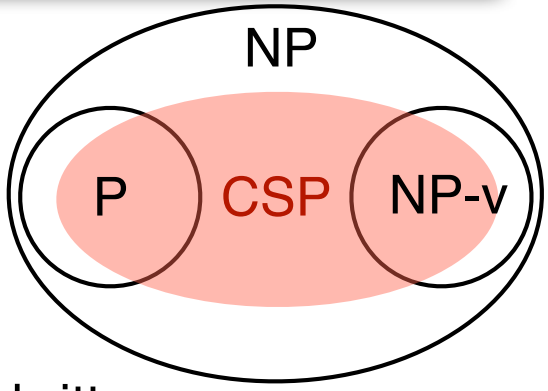
Probleme in NP und auch Probleme in P sind im Allgemeinen nicht unter homomorphen Urbildern abgeschlossen. Daher gilt:

Theorem

CSP $\subsetneq$ NP

T3.34

CSP und NP-Intermediate



Die Klasse CSP hat in vielerlei Hinsicht
bessere Eigenschaften als NP.

Ihre Analyse kann als ein erster (keineswegs trivialer) Schritt
auf dem Weg zu einem besseren Verständnis von NP dienen.

Insbesondere wurde 1993 die folgende Vermutung aufgestellt:

Feder-Vardi-Vermutung

Jedes CSP ist entweder in P oder NP-vollständig.

Die Vermutung impliziert also eine **Dichotomie zwischen P und NP:**
es gibt dann keine CSPs, die NP-intermediate sind.

Bis 2007 konnte sie nicht bewiesen werden,
aber es gab ernsthafte Fortschritte und partielle Resultate.

Partielle Resultate

Dichotomien für Strukturen mit eingeschränkter Größe

Satz von Schaefer (1978)

Sei $\mathfrak{A}$ eine relationale Struktur mit Wertebereich $A = \{0, 1\}$.
CSP($\mathfrak{A}$) ist in P, wenn eine der folgenden Bedingungen gilt:

1. Alle Relationen in $\mathfrak{A}$ enthalten das Tupel $(1, \ldots, 1)$.
2. Alle Relationen in $\mathfrak{A}$ enthalten das Tupel $(0, \ldots, 0)$.
3. Alle Relationen in $\mathfrak{A}$ sind durch Horn-Formeln definierbar.
4. Alle Relationen in $\mathfrak{A}$ sind durch Dual-Horn-Formeln definierbar.
5. Alle Relationen in $\mathfrak{A}$ sind durch 2-Formeln definierbar.
6. Alle Relationen in $\mathfrak{A}$ sind durch affine Formeln definierbar.

Anderenfalls ist CSP($\mathfrak{A}$) NP-vollständig.

T3.35

Dichotomie für Strukturen der Größe 3: Bulatov 2006

Partielle Resultate

Dichotomien für Strukturen mit eingeschränkten Relationen

Eine relationale Struktur ist *ungerichteter Graph*,
wenn sie nur ein einziges Relationssymbol E hat,
welches zudem binär ist und eine symmetrische Relation bezeichnet.

Theorem (Hell und Nešetřil 1990)

Sei $\mathfrak{A}$ ein ungerichteter Graph beliebiger Größe.

CSP($\mathfrak{A}$) ist in P, wenn $\mathfrak{A}$ 2-färbbar ist, und NP-vollständig sonst.

Feder und Vardi zeigen aber (1993),
dass eine Klassifikation von gerichteten Graphen (Symmetrie fällt weg)
ebenso schwer ist wie der allgemeine Fall.

2017: der Durchbruch

Feder-Vardi-Vermutung unabhängig bewiesen von

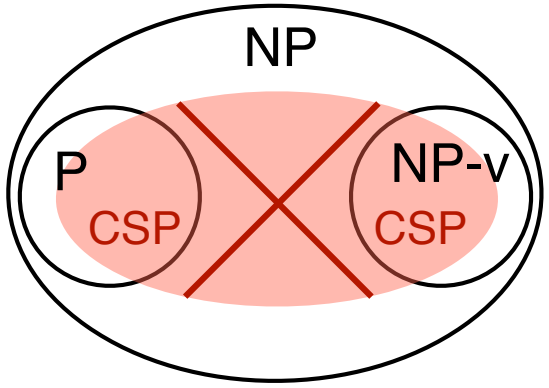
- (Rafiey, Kinne & Feder, `arXiv.org`, Feb. 2017)
  –> ist im Juli durch Gegenbeispiel entkräftet worden
- Bulatov, `FOCS` 2017
- Zhuk, `FOCS` 2017

Lance Fortnow (Computational Complexity Blog):

„I checked with experts in the field and at least one of these papers and more likely both ought to be correct.“

Theorem

Jedes CSP ist entweder in P oder NP-vollständig.



Übersicht Vorlesung

Kapitel 1: Einführung

Kapitel 2: Turingmaschinen

Kapitel 3: P vs. NP

Kapitel 4: Mehr Ressourcen, mehr Möglichkeiten?

Kapitel 5: Platzkomplexität

Kapitel 6: Schaltkreise

Kapitel 7: Orakel