Komplexitätstheorie

Kapitel 3: P vs NP

Einleitung

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

- Definitionen der Klassen
- Zusammenhang zwischen NP und Nichtdeterminismus
- Das Konzept der NP-Vollständigkeit
- Beispiele für NP-vollständige Probleme
- Einige Reflektionen zu P vs. NP
- Komplemente und co-NP

Kapitel 3

Definition von P und NP

Die Klasse P

Ein Ziel der Komplexitätstheorie:

> Identifizieren derjenigen Probleme, die (auch im worst case!) effizient lösbar sind.

Traditionell: "effizient lösbar (tractable)" = "in polynomieller Zeit lösbar"

(Polynom von beliebigem Grad!)

Entsprechende Komplexitätsklasse:

$$\mathsf{P} := \bigcup_{i \geq 1} \mathrm{DTIME}(n^i)$$ (oft auch ”PTIME”)

Die Klasse P

Rechtfertigung:

- Tabelle aus Kapitel 1:

Eingabegröße	1	2	3	4	5	6	7	8		20	128
n^2	1	4	9	16	25	36	49	64		400	16.384
2^n	2	4	8	16	32	64	128	256		1.048.576	RIESIG!

- Zeitkomplexität n^k mit sehr großem k ist auch bedenklich, aber: Polynome mit großem Grad sind fast nie notwendig!
- Nach Cobham-Edmonds These ist die Klasse P für alle Berechnungsmodelle identisch (wir können z.B o.B.d.A. mehrere Bänder verwenden)
- P ist abgeschlossen unter den üblichen Kompositionsoperationen auf Algorithmen

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, dessen Länge nicht viel größer ist als die 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

Der "Beweis" ist 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: Sequenzierung

Beispiel 2: Sequenzierung

Zur Erinnerung:

Definition (Single-Prozessor) Sequenzierung

Sei A eine Menge von *Aufgaben*, die durch $\prec$ partiell geordnet ist und $d : G \to \mathbb{N}$ Funktion, die Deadlines beschreibt.

Sequenzierung für A mit $k \in \mathbb{N}$ Verspätungen ist injektive Abbildung $\tau : G \to \{0, \ldots, |A| - 1\}$ so dass

1. $a \prec a'$ impliziert $\tau(a) < \tau(a')$;
2. für höchstens k Aufgaben $a \in A$ gilt $\tau(a) > d(a)$.

Beispiel 2: Sequenzierung

SEQ ist Menge aller Paare (A, k) so dass es Sequenzierung für A mit k Verspätungen gibt.

Der "Beweis" ist Sequenzierung selbst, repräsentiert als Folge von Aufgaben

Der Beweis ist einfach zu verifizieren: man kann offensichtlich in polynomieller Zeit prüfen, ob Bed. 1+2 aus Def. Sequenzierung erfüllt sind

Der Beweis hat Länge $|A|$, ist also kurz

Beispiel 3: Integer Programming

Definition Integer Programming

Sei V eine Menge von Variablen und G eine Menge von linearen Gleichungen

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

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

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

IPROG ist Menge aller Gleichunggsysteme G, für die Lösung existiert.

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)

Wenn G Gleichungssystem mit m Gleichungen, n Variablen und Konstanten beschränkt durch a, dann gibt es Lösung gdw. es eine Lösung aus dem Bereich $\{0, \ldots, n(ma)^{2m+1}\}$.

Wir verwenden als "kleine Beweise":

- 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 viele tausend solcher Probleme

NB: nicht alle entscheidbaren Probleme sind von dieser Art

z.B.: gegeben zwei reguläre Ausdrücke a_1, a_2, die zusätzlich zu $\cdot, \cup, \cdot^*$ auch $\cap$ benutzen dürfen, entscheide ob $L(a_1) = L(a_2)$ (Äquivalenzproblem)

Ist entscheidbar aber alle möglichen Beweise exponentiell groß

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 (also: 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, SEQ, IPROG sind in NP.

Kapitel 3

P vs NP

P vs NP

Theorem

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

Sehr unbefriedigend:

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

Es ist sogar unbekannt, ob P=NP

(oder ob P=ExpTime oder ob alle drei Klassen verschieden)

P vs NP

Intuitive Formulierung der P vs NP Frage:

Ist das Finden eines Beweises schwieriger als das Überprüfen?

Unsere Intuition sagt "Ja", also P ≠ NP!

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

Die P=NP Frage:

- ist eines der wichtigsten offenen Probleme der Informatik
- Lösung wird vom Clay Mathematics Institute mit US$1.000.000 belohnt (Millenium Prize)

P vs NP

Konsequenzen eines Beweises von

- P $=$ NP: dramatisch. Tausende bisher als sehr schwierig eingestufter Probleme wären dann effizient lösbar
  (z.B. CLIQUE, SEQ, IPROG)

- P $\neq$ NP: weniger dramatisch, aber wichtig zu wissen möglicherweise interessante Konsequenzen in der Kryptographie
  (dort: schwierige Probleme nützlich statt ärgerlich)

Kapitel 3

NP und nicht-deterministische TMs

Die Klasse NP

Die Klasse NP hat viele äquivalente Charakterisierungen:

- polynomielle Beweissysteme
- nicht-deterministische Turingmaschinen
- existentielle Logik zweiter Stufe (Fagins Theorem)
- randomisierte Beweissysteme (PCP Theorem)
- etc.

Nicht-deterministische TMs

Nicht-deterministische Turingmaschinen (kurz: 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 mind. eine Berechnung auf w akzeptierend

Nicht-deterministische TMs

NTMs

- entsprechen keinem real existierenden Rechner
- sind relevant wegen Ihres Zusammenhanges zu NP und anderen Komplexitätsklassen (gleich mehr)
- versteht man am besten über die Metapher des "Ratens":
  - NTM kann nicht-det. Wort w auf Band schreiben (”raten”)
  - Akzeptiert wird nur, wenn w so ist “wie gewünscht”
  - Da Akzeptanzbedingung von NTMs existentiell quantifiziert ist (“es gibt akzept. Berechnung”), ist folgende Vorstellung möglich: TM rät “richtig” wenn überhaupt möglich
  - Achtung: NTM kann nur etwas raten, dessen (Maximal)-Größe beschränkt ist.

Dadurch kann man NTM als DTM mit zusätzlicher Ratefunktion verstehen!

Nicht-deterministische 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 NTIME(t) ist definiert als Menge der Probleme

$$\{L \subseteq \Sigma^* \mid \text{es existiert } \mathcal{O}(t)\text{-zeitbeschränkte NTM } M \text{ mit } L(M) = L\}$$

Theorem (NTM Charakterisierung von NP)

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

Nicht-deterministische TMs

Klassischerweise ist diese Charakterisierung sogar die eigentliche Definition von NP

(aber nicht-deterministische Maschinen sind nicht sehr natürlich)

Konsequenzen der Charakterisierung:

- NP kann als das Äquivalent von P für NTMs angesehen werden
- Die P vs NP Frage ist dann:

  Kann eine nicht-deterministische Maschine in polynomieller Zeit mehr erreichen als eine deterministische Maschine?

Kapitel 3

NP-Härte, NP-Vollständigkeit

NP-Härte

Da es bisher nicht gelungen ist, zu zeigen, dass Probleme wie CLIQUE, SEQ, IPROG nicht in P sind, behilft man sich wie folgt:

Anstelle von absoluten Resultaten
(z.B. "Problem XYZ ist nicht in P")

betrachten wir relative Resultate
(z.B. "CLIQUE und SEQ sind gleich schwierig,
also CLIQUE in P gdw SEQ in P").

Besonders 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!)

Formalisierung von 1.: Begriff der NP-Härte, definiert über Reduktionen

Reduktionen

Reduktionen sind sehr wichtig in der Komplexitätstheorie:

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

Definition (Polynomielle) Reduktion

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

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

f heisst *polynomiell* wenn f in polynomieller Zeit berechenbar

Wenn Reduktion von L auf L' existiert: wir sagen L ist *reduzierbar* auf L' und schreiben $L \leq L'$

Polynomielle Reduzierbarkeit analog mit Notation $L \leq_p L'$.

Reduktionen

Intuitiv: $L \leq_p L'$ formalisiert "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:
   $L' \in$ P und $L \leq_p L'$ impliziert $L \in$ P;
2. NP ist abgeschlossen unter polynomiellen Reduktionen:
   $L' \in$ NP und $L \leq_p L'$ impliziert $L \in$ 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 = NP Problem ungelöst ist, wird NP-härte als "nicht effizient lösbar" gewertet, denn:

NP-Härte

Aber: warum sollte es überhaupt ein Problem geben, dass mindestens so schwierig ist wie jedes Problem in NP?

Theorem

NP-vollständige Probleme existieren.

Idee: definiere "universelles" Problem in NP, das alle Probleme in NP "enthält".

Noch überraschender: es gibt soger extrem viele und sehr natürliche NP-vollständige Probleme.

Kapitel 3

Cook's Theorem

NP-Härte

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

1. Zeigen, dass $L' \leq_p L$ für alle $L' \in$ NP

   Z.B. wie im vorigen Theorem: zeigen, dass das Wortproblem jeder polynomiell zeitbeschränkten NTM reduziert werden kann.

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

Theorem

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

SAT

Definition Formel, Wertzuweisung, Erfüllbar

Sei AV unendlich abzählbare Menge von *Aussagenvariablen*.
Menge der *aussagenlogischen Formeln* (kurz: AL-Formeln) ist kleinste Menge so dass:

- jedes $v \in$ AV ist AL-Formel
- wenn φ, ψ AL-Formeln, so auch $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$

Wertzuweisung (kurz: WZ) ist Abbildung π : AV $\rightarrow \{0, 1\}$.
WZ π *erfüllt* AL-Formel

- v wenn $\pi(v) = 1$;
- $\neg\varphi$, wenn π nicht φ erfüllt;
- $\varphi \wedge \psi$ wenn π sowohl φ als auch ψ erfüllt;
- $\varphi \vee \psi$ wenn π φ oder ψ erfüllt (oder beides).

AL-Formel φ ist erfüllbar, wenn es WZ π gibt, die φ erfüllt.

Cook's Theorem

Problem SAT ist Menge aller erfüllbaren AL-Formeln.

Theorem (Cook/Levin aka "Cooks Theorem")

SAT ist NP-vollständig.

In NP: $\{(\varphi, \pi) \mid \pi$ WZ für Variablen in φ, die φ erfüllt $\}$

ist polynomielles Beweissystem

NP-Härte:

Zeigen, dass Wortproblem jeder poly-zeitbeschränkten NTM polynomiell reduziert werden kann.

Cook's Theorem

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

Berechnung von M auf $w = a_0 \cdots a_{n-1}$ ist Matrix:

$\triangleright$	q_0, a_0	a_1	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\triangleright$	b	q, a_1	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\triangleright$	b	q', b'	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$0..p(n)$

$0..p(n)+1$

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

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

Cook's Theorem

Repräsentation der Matrix mittels Variablen:

- $B_{a,i,t}$: zum Zeipunkt t ist Zelle i mit a beschriftet;
  (Nummerierung von Zellen und Zeitpunkten beginnt mit 0, nicht 1)
- $K_{i,t}$: 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 $w = a_1 \cdots a_n$:

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

Cook's Theorem

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:

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

Zellen ohne Kopf ändern sich nicht:

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

Eingabe wird akzeptiert:

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

Cook's Theorem

Bandbeschriftung, Kopfposition, Zustand sind eindeutig und definiert:

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

$$\bigwedge_{t\leq p(n)} \bigvee Q \wedge \bigwedge_{t\leq p(n)} \bigvee_{i\leq p(n)+1} K_{i,t} \wedge \bigwedge_{t\leq p(n),i\leq p(n)+1} \bigvee \Gamma$$

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

Lemma

φ_w erfüllbar gdw. M akzeptiert w

SAT

SAT ist also NP-vollständig und damit nicht effizient lösbar, es sei denn P=NP

Trotzdem gibt es sehr effiziente SAT Solver:

- verfügen über sehr gute Heuristiken
- lösen Probleme aus der Praxis mit vielen Millionen Variablen in wenigen Sekunden
- werden via Reduktionen auch zum Lösen anderen Probleme in NP verwendet

Typische P-Probleme können aber noch schneller (und robuster) gelöst werden

Kapitel 3

Weitere NP-vollständige Problem

3SAT

Für NP-Härte eines Problems L:

es reicht, zu zeigen, dass SAT $\leq_p L$

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

Definition 3SAT

Literal ist Aussagenvariable oder deren Negation.
Klausel ist Disjunktion von Literalen
Formel in *Klauselform* ist Konjunktion von Klauseln
3-Klausel ist Klausel mit genau 3 Literalen
3-Formel ist Konjunktion von 3-Klauseln
3SAT ist die Menge aller erfüllbaren 3-Formeln.

3SAT ist offensichtlich in NP, da SAT in NP!

Einschub: NNF

Definition NNF

AL-Formel φ ist in *Negations-Normalform (NNF)* wenn Negation in φ nur direkt vor Variablen vorkommt (nicht vor zusammengesetzten Formeln!)

Zwei AL-Formeln φ und ψ sind *äquivalent* wenn für alle WZ π gilt:
π erfüllt φ gdw. π erfüllt ψ

Jede AL-Formel kann in polynomieller Zeit in äquivalente Formel in NNF gewandelt werden.

Erschöpfendes Anwenden folgender Regeln:

- $\neg\neg\varphi \longrightarrow \varphi$ (doppelte Negation)
- $\neg(\varphi \vee \psi) \longrightarrow \neg\varphi \wedge \neg\psi$ (de Morgan)
- $\neg(\varphi \wedge \psi) \longrightarrow \neg\varphi \vee \neg\psi$ (de Morgan)

3SAT

Theorem

3SAT ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von SAT

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

Idee:

- zuerst wandeln wir φ in NNF
- 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

CLIQUE

Theorem

CLIQUE ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

Gegeben 3-Formel φ, konstruiere in polynomieller Zeit Graph G und wähle $k \geq 1$ so dass φ erfüllbar gdw. G k-Clique enthält.

Zwei Literale φ und ψ sind *komplementär* gdw. sie dieselbe Variable enthalten, ein Literal positiv ist und eines negativ

Idee:

- G hat Knoten für jedes Vorkommen eines Literals in φ
- Kante zwischen zwei Literalen gdw. sie in unterschiedlichen Klauseln vorkommen und nicht komplementär sind
- Als k wähle Anzahl der Klauseln

3-Färbbarkeit

Viele Probleme auf Graphen sind NP-Vollständig, siehe VL "Algorithmen auf Graphen" von H-J Kreowski

Wir betrachten ein weiteres Beispiel

Definition 3-Färbbarkeit

Ungerichteter Graph $G = (V, E)$ ist *3-färbbar* gdw. 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.

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 einen Knoten für jedes Literal ein
- verwende die Farben “wahr”, “falsch” und “hilf”
- verbinde die Literale mit Kanten, um konsistent WZ zu erzwingen
- verwende Teilgraphen, um Erfülltsein jeder Klausel zu erzwingen

NP-Vollständigkeit

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

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

Im Buch "Computers and Intractability" von Garey und Johnson finden sich ca 300 NP-vollständige Probeme, insgesamt gibt es tausende!

Wir betrachten zwei weitere Beispiele:

- Integer Programming
- Sequenzierung

IPROG

Zur Erinnerung:

Definition Integer Programming

Sei V eine Menge von Variablen und G eine Menge von linearen Gleichungen

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

mit $x_1, \ldots, x_n \in V$, $c_1, \ldots, c_n \in \mathbb{N}$. und $\alpha \in V \cup \mathbb{N}$.
Lösung für G ist Abbildung $\tau : V \to \mathbb{N}$, so dass alle Gleichungen in G erfüllt sind.

IPROG ist Menge aller Gleichungssysteme G, für die 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
- Gleichung stellt sicher, dass diese Variablen konsistente Werte aus dem Bereich $\{0, 1\}$ erhalten
- Zusätzliche Gleichungen und numerische “Auffangvariablen” stellen sicher, dass jede Klausel wahres Literal enthält

Sequenzierung ist NP-Vollständig

Zur Erinnerung:

Definition (Single-Prozessor) Sequenzierung

Sei A eine Menge von *Aufgaben*, die durch $\prec$ partiell geordnet ist und $d : A \to \mathbb{N}$ Funktion, die Deadlines beschreibt.

Sequenzierung für A mit $k \in \mathbb{N}$ Verspätungen ist injektive Abbildung $\tau : A \to \{0, \ldots, |A| - 1\}$ so dass

1. $a \prec a'$ impliziert $\tau(a) < \tau(a')$;
2. für höchstens k Aufgaben $a \in A$ gilt $\tau(a) > d(a)$.

Theorem

Sequenzierung ist NP-vollständig.

Enthaltensein in NP schon gezeigt.

Sequenzierung ist NP-Vollständig

Härtebeweis durch Reduktion des Cliquenproblems:

Gegeben Graph G und Cliquengröße c, konstruiere in polynomieller Zeit Menge von Aufgaben A mit assoziiertem d und $\prec$ und Verspätungszahl k so dass G eine c-Clique hat gdw. es Sequenzierung für A mit k Verspätungen gibt.

Idee:

- verwende sowohl Knoten als auch Kanten als Aufgaben
- bestimme ”globale” deadline, vor der alle nicht verspäteten Kanten sequenziert werden müssen
- erzwinge mittels $\prec$, dass alle Kanten nach ihren Endpunkten sequenziert werden
- wähle globale Deadline und Anzahl Verspätungen so, dass nur die Knoten und Kanten einer Clique vor die deadline ”passen”

NP Vollständigkeit

NP-vollständige Probleme sind überall:

- Minesweeper. Gegeben den momentanen Stand eines Minesweeper Spieles (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 Manschaften), kann eine bestimmte Manschaft noch Meister werden?
- etc.

Kapitel 3

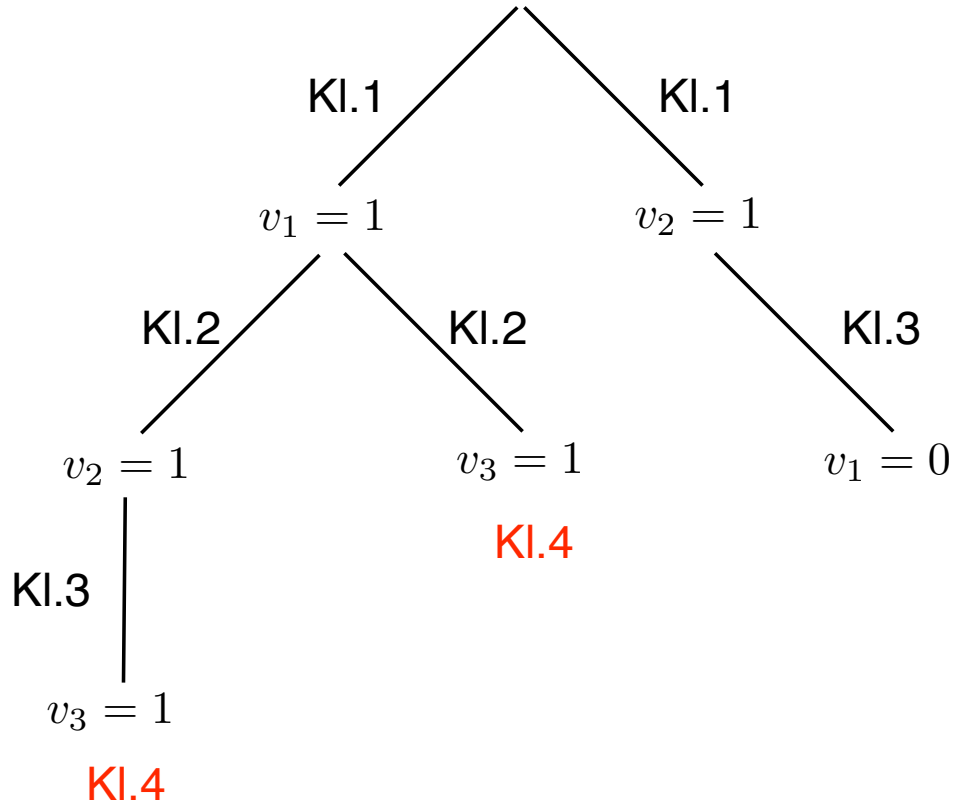
Reflektion zu NP-Härte

Kombinatorik

NP-harte Probleme sind oft "kombinatorisch", wie logische Puzzles

z.B. 3SAT:

$$(v_1 \vee v_2) \wedge (\neg v_1 \vee v_2 \vee v_3) \wedge (\neg v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_1 \vee \neg v_3)$$



Kombinatorik

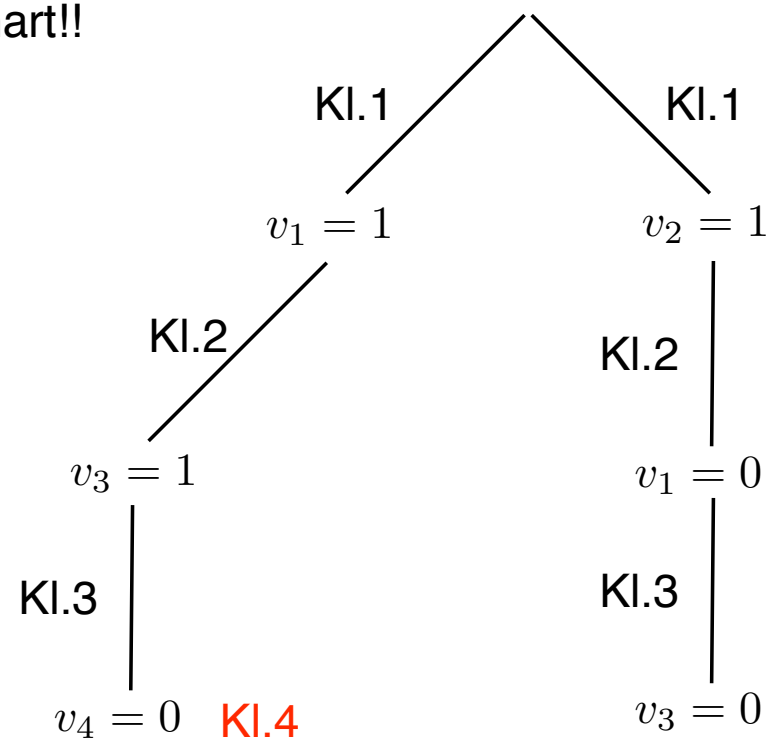
In einer solchen Situation sollte man versuchen, parallel

- NP-Härte zu zeigen
- einen polynomiellen Algorithmus zu finden

Denn: nicht alle "kombinatorisch aussehenden" Probleme sind auch wirklich NP-hart!!

z.B. 2SAT:

$$(v_1 \vee v_2) \wedge (\neg v_1 \vee v_3) \wedge (\neg v_3 \vee \neg v_4) \wedge (\neg v_1 \vee v_4)$$



2SAT

Andere Sicht auf 2-Klauseln und 3-Klauseln:

- 2-Klausel $(\ell_1 \vee \ell_2)$ entspricht Implikationen $\neg\ell_1 \to \ell_2$ und $\neg\ell_2 \to \ell_1$;
- 3-Klausel $(\ell_1 \vee \ell_2 \vee \ell_3)$ entspricht Implikationen

$$\begin{array}{ll} \neg\ell_1 \to \ell_2 \vee \ell_3 & \neg\ell_1 \wedge \neg\ell_2 \to \ell_3 \\ \neg\ell_2 \to \ell_1 \vee \ell_3 & \neg\ell_1 \wedge \neg\ell_3 \to \ell_2 \\ \neg\ell_3 \to \ell_1 \vee \ell_2 & \neg\ell_2 \wedge \neg\ell_3 \to \ell_1 \end{array}$$

Die Disjunktion im Kopf erfordert Auswahl!

Theorem

2SAT ist in P.

2SAT

Beweisskizze:

- Sei φ 2-Formel mit Variablen $v_1, \ldots, v_k$
- Betrachte φ als Menge von Implikationen:
  Klausel $(\ell \vee \ell')$ wird zu $(\neg\ell \to \ell')$ und $(\neg\ell' \to \ell)$
- Konstruiere gerichteten Graph $G_\varphi = (V_\varphi, E_\varphi)$ mit
  - $V_\varphi = \{v_1, \ldots, v_k, \neg v_1, \ldots, \neg v_k\}$
  - $E_\varphi = \{(\ell, \ell') \mid \ell \to \ell' \in \varphi\}$
- φ ist unerfüllbar gdw. es Variable v gibt so dass
  1. $\neg v$ ist in G_φ von v aus erreichbar und
  2. v ist in G_φ von $\neg v$ aus erreichbar.
- Der Graph kann in Polyzeit konstruiert werden und poly viele Erreichbarkeitsprobleme können in Polyzeit gelöst werden.

Bipartites Matching

Weiteres Beispiel für ein nur auf den ersten Blick schwieriges Problem

Definition Bipartiter Graph

Ein ungerichteter Graph $G = (V, E)$ ist *bipartit* wenn V in zwei Mengen V_1 und V_2 partitioniert ist und $\{v, v'\} \in E$ impliziert dass (i) $v \in V_1$ und $v' \in V_2$ oder (ii) $v' \in V_1$ und $v \in V_2$.

Definition Matching

Matching für Bipartiten Graphen $G = (V, E)$ ist Teilmenge $M \subseteq E$ so dass es für jeden Knoten $v \in V$ höchstens eine Kante $K \in E$ gibt mit $v \in K$.

Bipartites Matching

Meist interessiert man sich für Matchings mit maximaler Kardinalität
(Optimierungsproblem!)

Z.B. Zuordnung Maschinen - gleichzeitig auszuführende Aufgaben
("Heirats-Problem")

Problem MATCH: Paare (G, k) mit G bipartiter Graph und $k \geq 0$,
so dass es Matching M für G gibt mit $|M| \geq k$.

"Höchstens eine Kante" sieht nach Auswahl (und Kombinatorik) aus

Dennoch:

Theorem

MATCH ist in P

Beweis: siehe Standardlehrbücher zu Algorithmen

Kapitel 3

Einige weitere Themen rund um P und NP:

- Starke NP-Vollständigkeit
- Probleme, die weder NP-hart noch in P sind
- Isomorphie von NP-Problemen
- Komplemente und co-NP

Kapitel 3

Starke NP-Vollständigkeit

Starke NP-Vollständigkeit

Viele natürliche Probleme erhalten Zahlen als Eingabe:

z.B. CLIQUE und SEQ

Bei bisher betrachteten Problemen ist Zahlenwert beschränkt durch Größe der restlichen Eingabe:

- CLIQUE: Cliquengröße beschränkt durch Größe des Graphen
- SEQ: Anzahl Verspätungen und Deadlines beschränkt durch Anzahl Aufgaben

Das ist aber nicht immer so!

Rucksackproblem

- es soll 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 hat und Nutzen n_i.

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 Lösung existiert.

Rucksackproblem

Theorem

RP ist NP-Vollständig

Beweis Härte: polynomielle Reduktion von 3SAT

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

Idee:

- Verwende für jede Variable v_i zwei Gegenstände a_i und $\overline{a}_i$
- Ist v_i wahr ist, ist a_i im Rucksack aber $\overline{a}_i$ nicht
- Ist v_i falsch, so ist es genau andersherum
- Für jede Aufgabe 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 ●

Starke NP-Vollständigkeit

Werte von G, N und der g_i, n_i nicht durch andere Eingaben beschränkt

(RP ist ein sog. Problem auf großen Zahlen)

Beobachtung:

In Reduktion $3\text{SAT} \leq_p$ RP werden wirklich große Werte konstruiert:

$$G, N \geq 10^{k+\ell}$$

Natürliche Frage: ist RP für "kleine" Werte immernoch NP-hart?

Definition Stark NP-vollständig

Ein Problem L auf großen Zahlen heisst *stark NP-vollständig* wenn es ein Polynom p gibt, so dass L auf Eingaben, deren Zahlenwerte bei Eingabelänge n durch $p(n)$ beschränkt sind, NP-vollständig ist.

Wenn ein Problem NP-vollständig ist aber nicht stark NP-vollständig, dann kann es für manche Anwendungen immernoch effizient lösbar sein!

Starke NP-Vollständigkeit

Wenn P = NP, dann ist NP-Vollständigkeit = starke NP-Vollständigkeit

Theorem

Wenn P ≠ NP, dann ist RP nicht stark NP-Vollständig.

Idee:

- Wir nehmen polynomielle Schranke für Gewichtsgrenze G an
- Wir zeigen: wenn G polynomiell beschränkt, dann ist RP in P
- Ansatz: dynamisches Programmieren
- Verwende Teilprobleme RP(k, g), $1 \leq k \leq n$, $0 \leq g \leq G$
  wo nur die ersten k Gegenstände verwendet werden
  und die Gewichtsgrenze g ist.

Starke NP-Vollständigkeit

Einen Algorithmus wie diesen nennt man *pseudopolynomiell*

Starke NP-Vollständigkeit beweist man durch Reduktion eines NP-harten Problems unter Verwendung von kleinen Zahlen!

Beispiele für stark NP-Vollständige Probleme:

- Travelling Salesman (TSP),
  siehe VL "Algorithmen auf Graphen" von Hans-Jörg Kreowski
- 3-Partition
  siehe Garey & Johnson
- etc.

Kapitel 3

NP-Intermediate

NP Intermediate

Die Resultate mögen den Eindruck vermitteln, dass wenn P $\neq$ NP ist, dann für jedes Problem $L \in$ NP $\setminus$ P gilt: L ist NP-vollständig.

Dieser Eindruck ist nicht korrekt!

Definiere Komplexitätsklasse NPI (NP-Intermediate):

Menge aller Probleme $L \in$ NP $\setminus$ P, die nicht NP-vollständig sind.

Ladner's Theorem

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

Sogar:
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 $L_i \not\leq_p L_{i+1}$.

Zeigt, dass NP-Vollständigkeit auch noch wichtiges Konzept ist wenn P $\neq$ NP!

NP Intermediate

Im Beweis von Ladner's Theorem wird "unnatürliches" Problem konstruiert:

Starte mit SAT, lasse genug "Ja"-Instanzen weg so dass

- jede mögliche polynomielle Reduktion von SAT auf das resultierende Problem fehlschlägt
- jeder mögliche polynomielle Algorithmus für das resultierende Problem fehlschlägt.

Bisher konnte kein natürliches Problem in NPI identifiziert werden
(nicht mal unter der Annahme P ≠ NP)

Kandidaten:

- Graph Isomorphismus. Gegeben ungerichtete Graphen G_1, G_2, kann man die Knoten in G_2 umbenennen so dass $G_1 = G_2$?
- Integer Faktorisierung. Gegeben $i, j \in \mathbb{N}$, hat i einen Faktor kleiner als j?

Kapitel 3

Isomorphie von NP-Problemen

Isomorphievermutung

NP-vollständige Probleme scheinen sich alle 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:

- Injektivität nicht gewährleistet
  z.B. 3SAT $\leq_p$ CLIQUE: konsistentes Austauschen von Variablen ändert konstruierten Graph/Cliquenzahl 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 genau?

Isomorphievermutung

Wir

- fordern Injektivität und Surjektivität
- verschmelzen 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 sind L' und L *p-isomorph*.

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

Isomorphievermutung

Vermutung (Berman/Hartmanis)

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 $P \neq NP$.

Alle bekannten NP-vollständigen Probleme sind jedoch paarweise p-isomorph

Isomorphievermutung

Grundlage für injektive + umkehrbare Reduktionen: Padding Funktionen

Idee: Kodiere Eingabewort in die von der Reduktion berechnete Instanz

Definition Padding Funktion

Sei $L \subseteq \Sigma^*$. Funktion pad : $(\Sigma \times \Gamma) \to \Sigma^*$ ist *Padding Funktion* für L wenn:

1. pad(x, y) kann in Zeit poly$(|x| + |y|)$ berechnet werden
2. für alle $x, y \in \Sigma^*$: pad$(x, y) \in L$ gdw. $x \in L$
3. für alle $x, y \in \Sigma^*$: $|\text{pad}(x, y)| > |x| + |y|$
4. Gegeben pad(x, y) kann y in Polyzeit berechnet werden

Theorem

Wenn $L' \leq_p L$ und es eine Padding Funktion für L gibt, dann gibt es injektive Reduktion f von L' auf L mit f^{-1} berechenbar in Polyzeit.

Isomorphievermutung

Die Reduktion aus vorigem Theorem ist fast p-Isomorphismus:

- injektiv und Umkehrung berechenbar in Polyzeit
- aber nicht unbedingt surjektiv

Um Surjektivität zu erreichen kombiniert man

- eine solche Reduktion f von L' nach L und
- g^{-1} für eine solche Reduktion g von L nach L'

Theorem

Wenn $L \subseteq \Sigma^*$ und $L' \subseteq \Gamma^*$ NP-vollständig und es Padding Funktionen für L und L' gibt, dann sind L und L' p-Isomorph.

Beweis: Papadimitriou

Kapitel 3

Komplemente und co-NP

Motivation

Subtile Frage:

Sei $\overline{3F}$ die Menge aller nicht 3-färbbaren ungerichteten Graphen.
In welcher Komplexitätsklasse ist $\overline{3F}$?

Um in NP zu sein, müsste es polynomielles Beweissystem geben.

Aber was ist polynomieller Beweis dafür, dass Graph nicht 3-färbbar?

Z.B. Menge aller Färbungen mit 3 Farben; exponentiell viele!

Es ist nicht bekannt, ob es für $\overline{3F}$ polynomielles Beweissystem gibt!

Komplement

Definition Komplement

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

Komplexitätsklasse $\mathcal{C}$ ist *abgeschlossen unter Komplement* wenn:

$$L \in \mathcal{C} \text{ impliziert } \overline{L} \in \mathcal{C}$$

Z.B. $\overline{\text{3F}}$

$\overline{\text{SAT}}$ ist die Menge aller unerfüllbaren AL-Formeln

$\overline{\text{CLIQUE}}$ ist die Menge aller Paare (G, k) so dass G keine k-Clique hat

Lemma

Deterministische Zeitkomplexitätsklassen wie DTIME(f) und P sind abgeschlossen unter Komplement.

Also z.B. $\overline{\text{GAP}} \in$ P und $\overline{\text{2SAT}} \in$ P

Komplement

Dieses Argument schlägt fehl für nicht-deterministische TMs,

funktioniert also nicht für die alternative Charakterisierung von NP!

Definition Komplementklassen

Wenn $\mathcal{C}$ Komplexitätsklasse, dann co-$\mathcal{C} := \{\overline{L} \mid L \in \mathcal{C}\}$.

Wegen P=co-P wird co-P nicht betrachtet, co-NP aber sehr wohl!

Z.B. $\overline{\text{3F}}, \overline{\text{SAT}}, \overline{\text{CLIQUE}} \in$ co-NP nach Definition.

Definition Gültigkeit in Aussagenlogik

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

Lemma

Gültigkeit ist in co-NP.

Vollständigkeit

Härte und Vollständigkeit sind definiert wie für NP:

Definition co-NP-Härte, co-NP-Vollständigkeit

Problem L ist

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

Lemma

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

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

Lemma

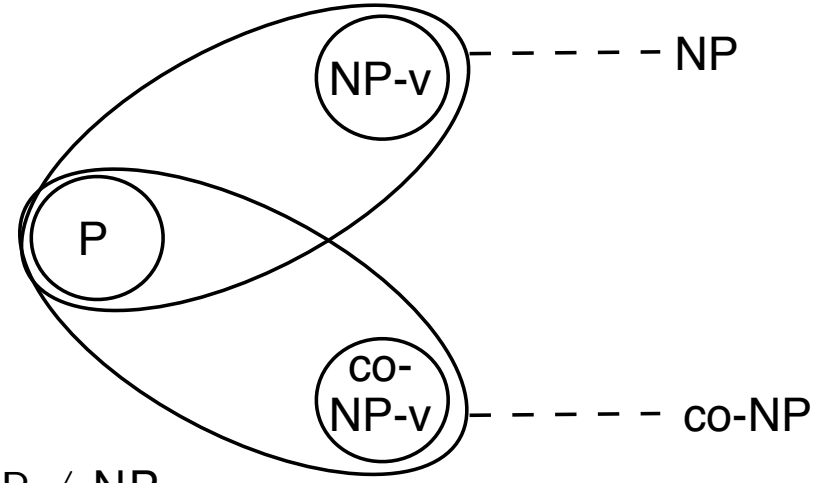
Gültigkeit ist co-NP-vollständig.

P vs NP vs co-NP

Leicht zu sehen: P $\subseteq$ co-NP

Es wird vermutet, dass

1. NP $\neq$ co-NP
2. NP $\cap$ co-NP $\neq$ P



Beide Vermutungen implizieren P $\neq$ NP
(aber nicht umgekehrt)

Lemma

Wenn NP $\neq$ co-NP und $L \in$ NP $\cap$ co-NP, dann ist L nicht NP-vollständig.

Es sind also alle Probleme in NP $\cap$ co-NP in P oder NP-Intermediate!

Kandidat: Integer Faktorisierung ist in NP und co-NP