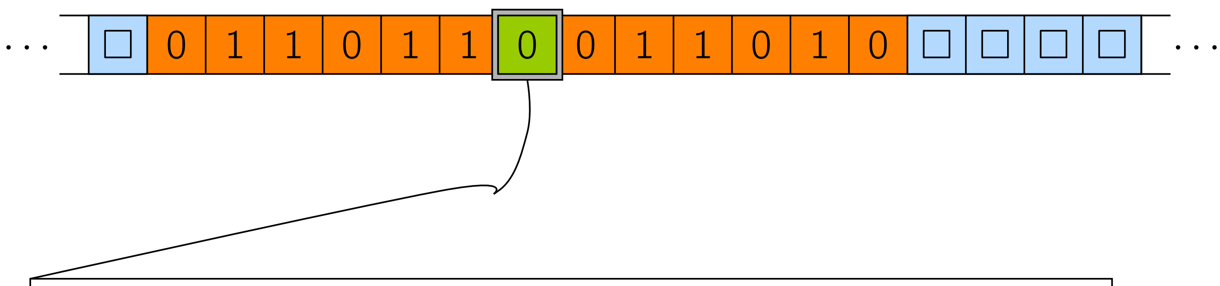
Vorlesung 14
Die Klasse NP und polynomielle Reduktionen

Wdh.: Nichtdeterministische Turingmaschine (NTM)



δ	0	1	B
q_0	$\{(q_0, B, R), (q_1, B, R)\}$	{reject}	{reject}
q_1	{reject}	$\{(q_1, B, R), (q_2, B, R)\}$	{reject}
q_2	{reject}	{reject}	{accept}

Wdh.: Komplexitätsklassen

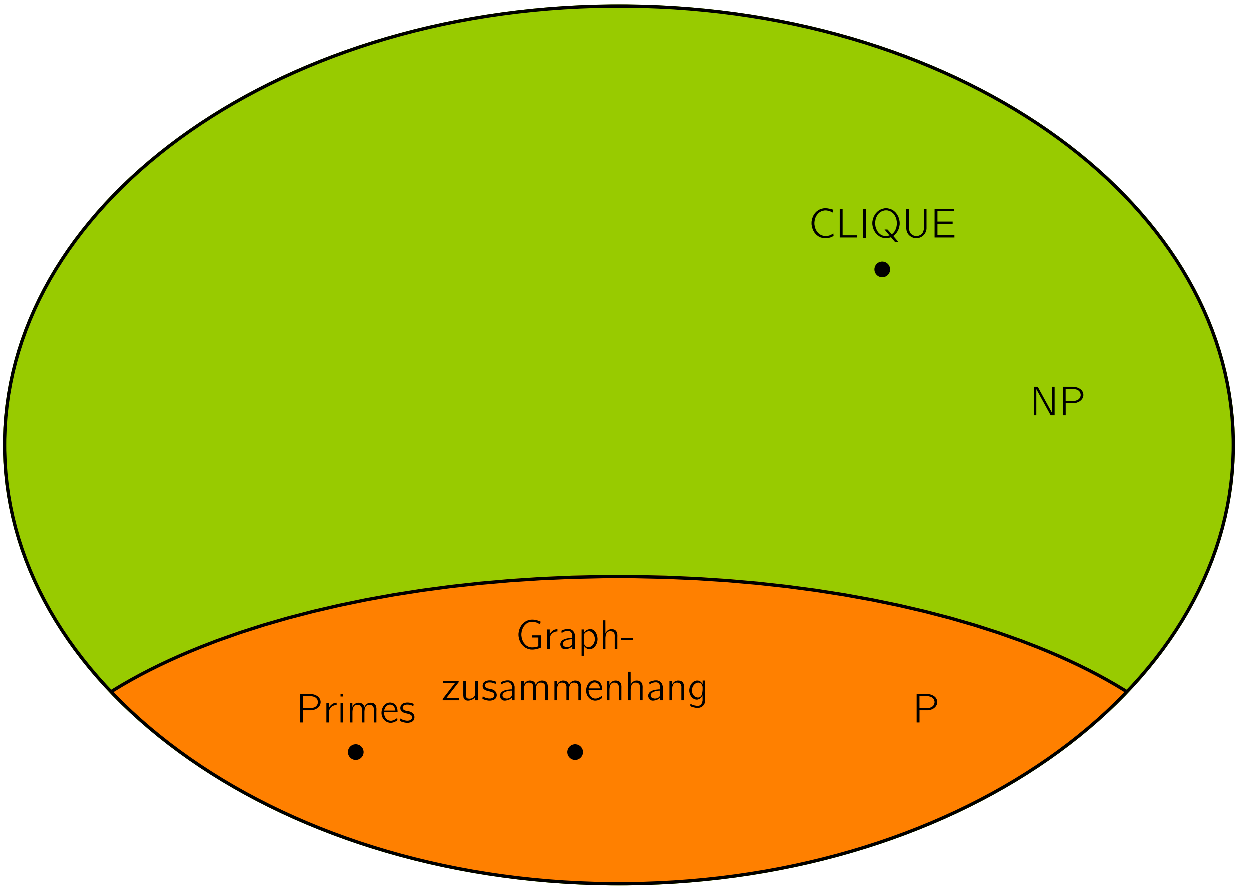
Definition (Komplexitätsklassen)

- P ist die Klasse der Entscheidungsprobleme, für die es einen Polynomialzeitalgorithmus gibt.
- NP ist die Klasse der Entscheidungsprobleme, die durch eine NTM M erkannt werden, deren worst case Laufzeit $t_M(n)$ polynomiell beschränkt ist.
- EXPTIME ist die Klasse der Entscheidungsprobleme L, für die es ein Polynom q gibt, so dass sich L auf einer DTM mit Laufzeitschranke $2^{q(n)}$ berechnen lässt.

Satz

$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXPTIME}$

Die Komplexitätslandschaft



Warnung: Dieser Abbildung liegt die Annahme $\mathrm{P} \neq \mathrm{NP}$ zu Grunde.

Optimierungsprobleme und ihre Entscheidungsvarianten

- Beim Rucksackproblem (KP) ist eine Menge von N Objekten mit Gewichten $w_1, \dots, w_N$ und Nutzenwerten $p_1, \dots, p_N$ gegeben.
- Des Weiteren ist eine Gewichtsschranke b gegeben.
- Wir suchen eine Teilmenge K der Objekte, so dass die Objekte aus K in einen Rucksack mit Gewichtsschranke b passen und dabei der Nutzen maximiert wird.

Problem (Rucksackproblem, Knapsack Problem – KP)

Eingabe: $b \in \mathbb{N}$, $w_1, \dots, w_N \in \{1, \dots, b\}$, $p_1, \dots, p_N \in \mathbb{N}$

zulässige Lösungen: $K \subseteq \{1, \dots, N\}$, *so dass* $\sum_{i \in K} w_i \leq b$

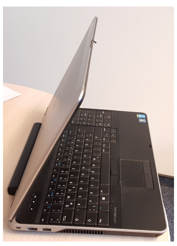
Zielfunktion: *Maximiere* $\sum_{i \in K} p_i$

Entscheidungsvariante: $p \in \mathbb{N}$ sei gegeben. Gibt es eine zulässige Lösung mit Nutzen mindestens p?

Rucksackproblem – Beispiel

maximal 10 kg

$p_1 = 20$
$w_1 = 6.5$ kg

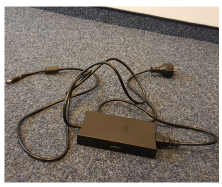


$p_2 = 200$
$w_2 = 3$ kg

$p_3 = 0.01$
$w_3 = 0.1$ kg

$p_4 = 5$
$w_4 = 1$ kg

$p_5 = 0.1$
$w_5 = 0.1$ kg



$p_6 = 30$
$w_6 = 1$ kg

Optimierungsprobleme und ihre Entscheidungsvarianten

Beim Bin Packing Problem suchen wir eine Verteilung von N Objekten mit Gewichten $w_1, \dots, w_N$ auf eine möglichst kleine Anzahl von Behältern mit Gewichtskapazität jeweils b.

Problem (Bin Packing Problem – BPP)

Eingabe: $b \in \mathbb{N}$, $w_1, \dots, w_N \in \{1, \dots, b\}$

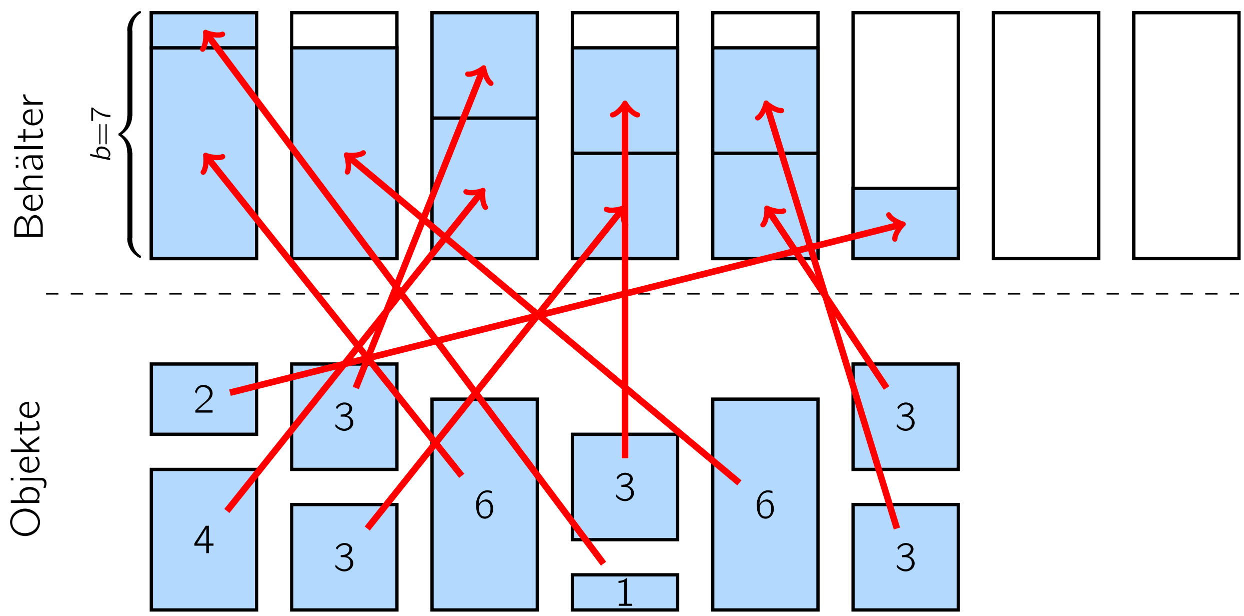
zulässige Lösungen: $k \in \mathbb{N}$ *und Funktion* $f : \{1, \dots, N\} \to \{1, \dots, k\}$,

$$\text{so dass } \forall i \in \{1, \dots, k\} : \sum_{j \in f^{-1}(i)} w_j \le b$$

Zielfunktion: *Minimiere* k *(= Anzahl Behälter)*

Entscheidungsvariante: $k \in \mathbb{N}$ sei gegeben. Passen die Objekte in k Behälter?

Bin Packing Problem – Beispiel



Eine Lösung die $k = 6$ Behälter verwendet

Optimierungsprobleme und ihre Entscheidungsvarianten

Beim Traveling Salesperson Problem (TSP) ist ein vollständiger Graph auf N Knoten (Orten) mit Kantengewichten (Kosten) gegeben. Gesucht ist eine Rundreise (ein Hamiltonkreis, eine Tour) mit kleinstmöglichen Kosten.

Problem (Traveling Salesperson Problem – TSP)

Eingabe: $c(i,j) \in \mathbb{N}$ *für* $i,j \in \{1,\ldots,N\}$ *mit* $c(j,i) = c(i,j)$

zulässige Lösungen: *Permutationen* π *auf* $\{1,\ldots,N\}$

Zielfunktion: *Minimiere* $c(\pi(N),\pi(1)) + \sum_{i=1}^{N-1} c(\pi(i),\pi(i+1))$

Entscheidungsvariante: $b \in \mathbb{N}$ ist gegeben. Gibt es eine Tour der Länge höchstens b?

Traveling Salesperson Problem – Beispiel

Man finde eine kürzeste Rundreise durch die größten deutschen Städte und zurück zum Ausgangsort.

Optimierungsproblem versus Entscheidungsproblem

- Mit Hilfe eines Algorithmus, der ein Optimierungsproblem löst, kann man die Entscheidungsvariante lösen. (Wie?)
- Häufig funktioniert auch der umgekehrte Weg. Wir illustrieren dies am Beispiel von KP.
- In den Übungen zeigen wir dasselbe auch für TSP und BPP.

Satz

Wenn die Entscheidungsvariante von KP in polynomieller Zeit lösbar ist, dann auch die Optimierungsvariante.

Beweis: Entscheidungsvariante $\to$ Zwischenvariante

Zwischenvariante: Gesucht ist nicht eine optimale Lösung, sondern nur der optimale Zielfunktionswert.

Polynomialzeitalgorithmus für die Zwischenvariante

Wir verwenden eine Binärsuche mit folgenden Parametern:

- Der minimale Profit ist 0. Der maximale Profit ist $P := \sum_{i=1}^{N} p_i$.
- Wir finden den optimalen Zielfunktionswert durch Binärsuche auf dem Wertebereich $\{0, \ldots, P\}$.
- Sei A ein Polynomialzeitalgorithmus für die Entscheidungsvariante von KP.
- In jeder Iteration verwenden wir Algorithmus A, der uns sagt in welche Richtung wir weitersuchen müssen.

Beweis: Entscheidungsvariante → Zwischenvariante

Die Anzahl der Iterationen der Binärsuche ist $\lceil \log(P+1) \rceil$.

Diese Anzahl müssen wir in Beziehung zur Eingabelänge n setzen.

Untere Schranke für die Eingabelänge:

- Die Kodierungslänge von $a \in \mathbb{N}$ ist $\kappa(a) := \lceil \log(a+1) \rceil$.
- Die Funktion κ ist subadditiv, d.h., für alle $a, b \in \mathbb{N}$ gilt $\kappa(a+b) \leq \kappa(a) + \kappa(b)$.
- Die Eingabelänge n ist somit mindestens

$$\sum_{i=1}^{N} \kappa(p_i) \;\geq\; \kappa\left(\sum_{i=1}^{N} p_i\right) \;=\; \kappa(P) \;=\; \lceil \log(P+1) \rceil \;.$$

Also reichen n Aufrufe von A um den optimalen Zielfunktionswert zu bestimmen.

Beweis: Zwischenvariante → Optimierungsvariante

Aus einem Algorithmus B für die Zwischenvariante konstruieren wir jetzt einen Algorithmus C für die Optimierungsvariante.

Algorithmus C

1. $K := \{1, \ldots, N\}$;
2. $p := B(K)$;
3. **for** i := 1 **to** N **do**
   if $B(K \setminus \{i\}) = p$ **then** $K := K \setminus \{i\}$;
4. **Ausgabe** K.

Laufzeit: $N+1$ Aufrufe von Algorithmus B, also polynomiell beschränkt, falls die Laufzeit von B polynomiell beschränkt ist.

Beweis: Zwischenvariante $\to$ Optimierungsvariante

Korrektheit:

Es gilt die Schleifeninvariante $B(K) = p$.

Für die ausgegebene Menge K gilt somit $B(K) = p$.

Aber ist die ausgegebene Menge K auch zulässig?

- Zum Zweck des Widerspruchs nehmen wir an, dass $\sum_{i \in K} w_i > b$.
- Dies bedeutet, dass nicht alle Objekte in den Rucksack passen.
- Dann gibt es $i \in K$, so dass $B(K \setminus \{i\}) = p$.
- Dies steht aber im Widerspruch dazu, dass der Algorithmus das Objekt i nicht aus K gestrichen hat.
- Also gilt $\sum_{i \in K} w_i \leq b$ und somit ist K zulässig.

Alternative Charakterisierung der Klasse NP

Satz

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann in NP, *wenn es einen Polynomialzeitalgorithmus V (einen sogenannten Verifizierer) und ein Polynom p mit der folgenden Eigenschaft gibt:*

$$x \in L \quad \Leftrightarrow \quad \exists y \in \{0,1\}^*, |y| \leq p(|x|) : V \textit{ akzeptiert } y\#x.$$

Von der NTM zu Zertifikat & Verifizierer

Es sei $L \in \mathrm{NP}$ eine Sprache.

- Sei M eine NTM, die L in polynomieller Zeit erkennt.
- Die Laufzeit von M sei beschränkt durch ein Polynom q.
- O.B.d.A. sehe die Überführungsrelation von δ immer genau zwei Übergänge vor, die wir mit 0 und 1 bezeichnen.

Konstruktion von Zertifikat und Verifizierer:

- Für die Eingabe $x \in L$ beschreibe $y \in \{0,1\}^{q(|x|)}$ den Pfad von M auf einem akzeptierenden Rechenweg.
- Wir verwenden y als Zertifikat.
- Der Verifizierer V erhält als Eingabe $y\#x$ und simuliert einen Rechenweg der NTM M für die Eingabe x.

Beweis: Von der NTM zu Zertifikat & Verifizierer

Korrektheit der Konstruktion:

- Gemäß Konstruktion gilt

$$\begin{aligned} x \in L \quad &\Leftrightarrow \quad M \text{ akzeptiert } x \\ &\Leftrightarrow \quad \exists y \in \{0,1\}^{q(|x|)} : V \text{ akzeptiert } y\#x. \end{aligned}$$

- Der Verifizierer kann die durch das Zertifikat y beschriebene Rechnung mit polynomiellem Zeitverlust simulieren.
- Somit erfüllen y und V die im Satz geforderten Eigenschaften.

Beweis: Von Zertifikat & Verifizierer zur NTM

Gegeben:

Verifizierer V mit polynomieller Laufzeitschranke und Polynom p mit der Eigenschaft:

$$x \in L \quad \Leftrightarrow \quad \exists y \in \{0,1\}^*, |y| \leq p(|x|) : V \text{ akzeptiert } y\#x.$$

Konstruktion der NTM:

1. M rät das Zertifikat $y \in \{0,1\}^*, |y| \leq p(|x|)$.
2. M führt V auf $y\#x$ aus und akzeptiert genau dann, wenn V akzeptiert.

Beweis: Von Zertifikat & Verifizierer zur NTM

Korrektheit der Konstruktion:

- M erkennt die Sprache L, weil gilt

$$\begin{aligned} x \in L \quad &\Leftrightarrow \quad \exists y \in \{0,1\}^*, |y| \leq p(|x|) : V \text{ akzeptiert } y\#x \\ &\Leftrightarrow \quad \text{Es gibt einen akzeptierenden Rechenweg für } M \\ &\Leftrightarrow \quad M \text{ akzeptiert } x. \end{aligned}$$

- Die Laufzeit von M ist polynmiell beschränkt, denn
  - die Laufzeit von Schritt 1 (Zertifikat raten) entspricht der Länge des Zertifikats, und
  - die Laufzeit von Schritt 2 (Zertifikat verifizieren) entspricht der Laufzeit des Verifizierers.

$\square$

Zertifikat & Verifizerer für KP, BPP und TSP

Satz

Die Entscheidungsvarianten von KP, BPP und TSP sind in NP.

Beweis:

Die Entscheidungsvariante eines Optimierungsproblems hat einen natürlichen Kandidaten für ein Zertifikat, nämlich zulässige optimale Lösungen.

Es muss allerdings gezeigt werden, dass

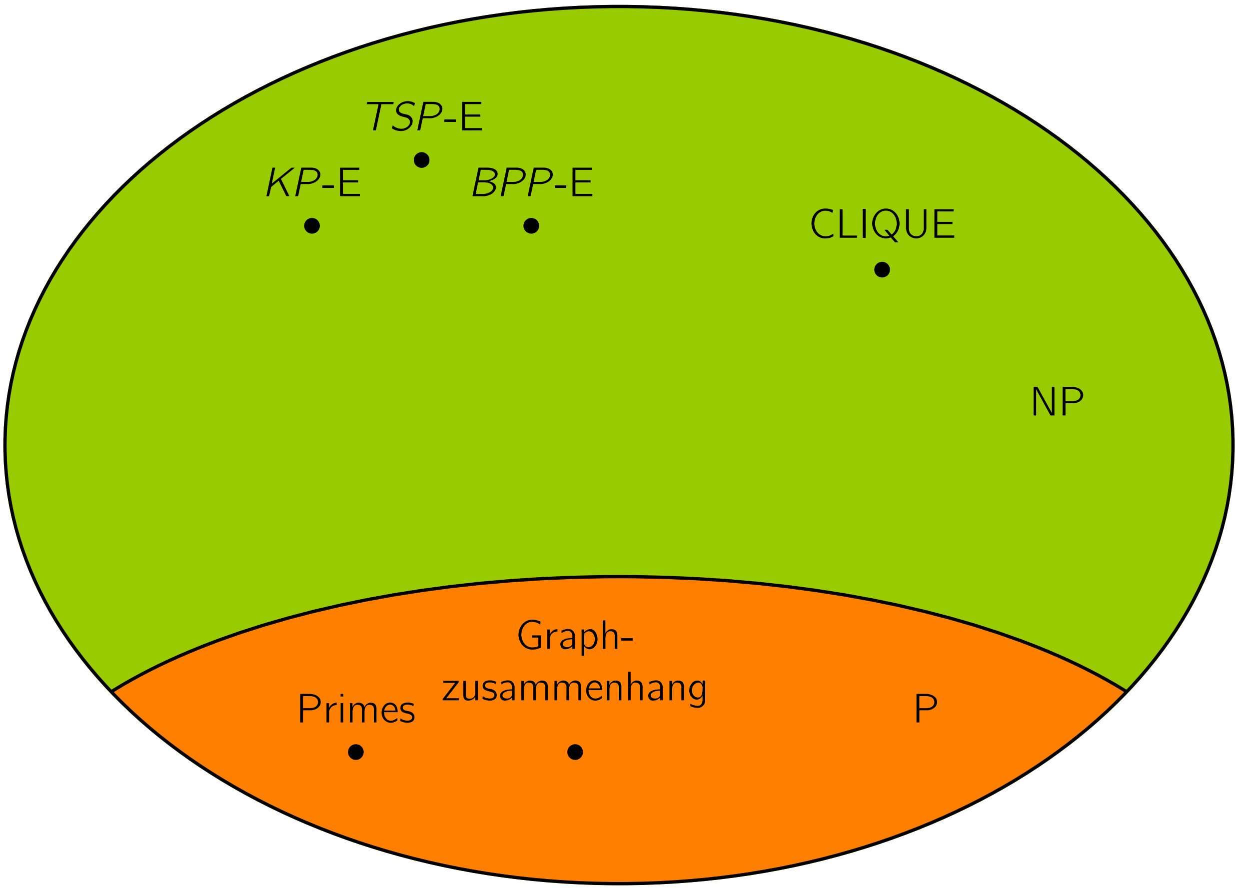
- diese Lösungen eine in der Eingabelänge polynomiell beschränkte Kodierungslänge haben, und
- ihre Zulässigkeit durch einen Polynomialzeitalgorithmus überprüft werden kann.

Zertifikat & Verifizerer für KP, BPP und TSP

- KP: Die Teilmenge $K \subseteq \{1, \ldots, N\}$ kann mit N Bits kodiert werden. Gegeben K, kann die Einhaltung von Gewichts- und Nutzenwertschranke in polynomieller Zeit überprüft werden.
- BPP: Die Abbildung $f\colon \{1, \ldots, N\} \to \{1, \ldots, k\}$ kann mit $O(N \log k)$ Bits kodiert werden. Gegeben f, kann die Einhaltung der Gewichtsschranken in polynomieller Zeit überprüft werden.
- TSP: Für die Kodierung einer Permutation π werden $O(N \log N)$ Bits benötigt. Es kann in polynomieller Zeit überprüft werden, ob die durch π beschriebene Rundreise die vorgegebene Kostenschranke b einhält.

$\square$

Die Komplexitätslandschaft



Warnung: Dieser Abbildung liegt die Annahme P $\neq$ NP zu Grunde.

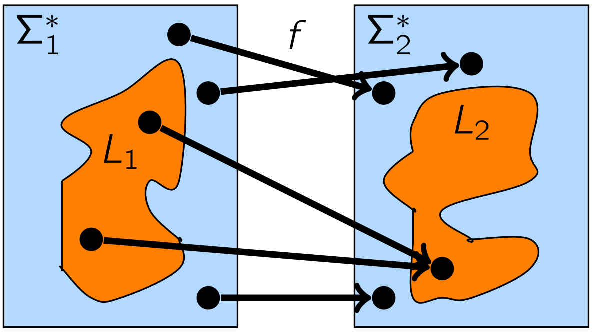
Polynomielle Reduktionen

Definition (Polynomielle Reduktion)

L_1 und L_2 seien zwei Sprachen über Σ_1 bzw. Σ_2. Dann heißt L_1 polynomiell reduzierbar auf L_2, wenn es eine Reduktion von L_1 nach L_2 gibt, die in polynomieller Zeit berechenbar ist. Wir schreiben $L_1 \leq_p L_2$.

D.h. $L_1 \leq_p L_2$ gilt genau dann, wenn es eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den folgenden Eigenschaften gibt:

- f ist in polynomieller Zeit berechenbar
- $\forall x \in \Sigma_1^*: x \in L_1 \Leftrightarrow f(x) \in L_2$



Polynomielle Reduktionen (Forts.)

Lemma

Angenommen $L_1 \leq_p L_2$, dann gilt: $L_2 \in P \Rightarrow L_1 \in P$.

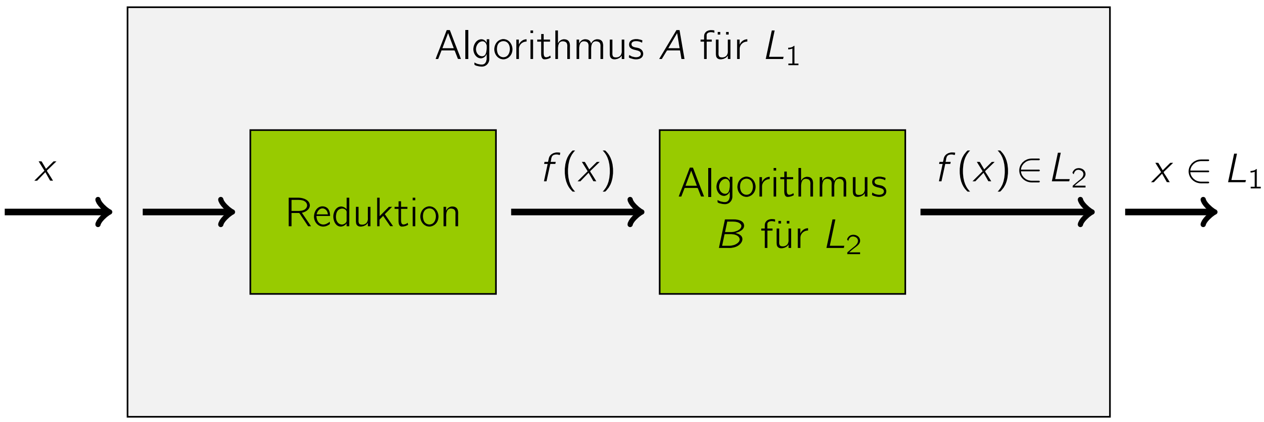
Beweis: Die Reduktion f habe die polynomielle Laufzeitschranke $p(\cdot)$.
Sei B ein Algorithmus für L_2 mit polynomielle Laufzeitschranke $q(\cdot)$.

Algorithmus A für L_1:

1. Berechne $f(x)$.
2. Simuliere Algorithmus B für L_2 auf $f(x)$ und übernehme die Antwort.

Schritt 1 hat Laufzeit höchstens $p(|x|)$. Schritt 2 hat Laufzeit höchstens $q(|f(x)|) \leq q(p(|x|) + |x|)$. $\square$

Polynomielle Reduktionen – Illustration



COLORING $\leq_p$ SAT

Mit dem Reduktionsprinzip ist es möglich, Probleme unterschiedlichster Art aufeinander zu reduzieren. Wir demonstrieren dies an einem Beispiel.

Problem (Knotenfärbung – COLORING)

Eingabe: Graph $G = (V, E)$, Zahl $k \in \{1, \dots, |V|\}$
Frage: Gibt es eine Färbung $c\colon V \to \{1, \dots, k\}$ der Knoten von G mit k Farben, so dass benachbarte Knoten verschiedene Farben haben, d.h. $\forall e = \{u, v\} \in E : c(u) \neq c(v)$?

Problem (Erfüllbarkeitsproblem / Satisfiability – SAT)

Eingabe: Aussagenlogische Formel φ in KNF
Frage: Gibt es eine erfüllende Belegung für φ?

Zwei Beispiele zum Erfüllbarkeitsproblem

SAT-Beispiel 1:

$$\varphi = (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_2 \vee x_3 \vee x_4)$$

φ ist erfüllbar, denn $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$ ist eine erfüllende Belegung.

SAT-Beispiel 2:

$$\varphi' = (x_1 \vee x_2) \wedge (\bar{x}_2 \vee x_1) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_1)$$

φ' ist nicht erfüllbar. (Warum?)

Eine polynomielle Reduktion: COLORING $\leq_p$ SAT

Satz

COLORING $\leq_p$ SAT.

Beweis:

Wir beschreiben eine polynomiell berechenbare Funktion f, die eine Eingabe (G, k) für das COLORING-Problem auf eine Formel φ für das SAT-Problem abbildet, mit der Eigenschaft

$$G \text{ hat eine } k\text{-Färbung} \;\Leftrightarrow\; \varphi \text{ ist erfüllbar} .$$

Beispiel einer polyn. Reduktion: COLORING $\leq_p$ SAT

Beschreibung der Funktion f:

Die Formel φ hat für jede Knoten-Farb-Kombination (v, i), $v \in V$, $i \in \{1, \ldots, k\}$ eine Variable x_v^i. Die Formel für (G, k) lautet

$$\varphi \;=\; \bigwedge_{v \in V} \underbrace{(x_v^1 \vee x_v^2 \vee \ldots \vee x_v^k)}_{\text{Knotenbedingung}} \wedge \bigwedge_{\{u,v\} \in E} \; \bigwedge_{i \in \{1,\ldots,k\}} \underbrace{(\bar{x}_u^i \vee \bar{x}_v^i)}_{\text{Kantenbedingung}} \quad .$$

Anzahl der Literale $= O(k \cdot |V| + k \cdot |E|) = O(|V|^3)$.

Die Länge der Formel ist somit polynomiell beschränkt und die Formel kann in polynomieller Zeit konstruiert werden.

Aber ist die Konstruktion auch korrekt?

Beispiel einer polyn. Reduktion: COLORING $\leq_p$ SAT

Korrektheit:

Zu zeigen: G hat eine k-Färbung $\Rightarrow$ φ ist erfüllbar

- Sei c eine k-Färbung für G.
- Für jeden Knoten v mit $c(v) = i$ setzen wir $x_v^i = 1$ und alle anderen Variablen auf 0.
- Knotenbedingung: $(x_v^1 \vee x_v^2 \vee \ldots \vee x_v^k)$ ist erfüllt.
- Kantenbedingung: Für jede Farbe i und jede Kante $\{u, v\}$ gilt $\bar{x}_u^i \vee \bar{x}_v^i$, denn sonst hätten u und v beide die Farbe i.
- Damit erfüllt diese Belegung die Formel φ.

Beispiel einer polyn. Reduktion: COLORING $\leq_p$ SAT

Zu zeigen: φ ist erfüllbar $\Rightarrow$ G hat eine k-Färbung

- Fixiere eine beliebige erfüllende Belegung für φ.
- Wegen der Knotenbedingung gibt es für jeden Knoten v mindestens eine Farbe i mit $x_v^i = 1$.
- Für jeden Knoten wähle eine beliebige derartige Farbe aus.
- Sei $\{u, v\} \in E$. Wir behaupten: $c(u) \neq c(v)$.
- Zum Widerspruch nehmen wir an, $c(u) = c(v) = i$. Dann wäre $x_u^i = x_v^i = 1$ und die Kantenbedingung $\bar{x}_u^i \vee \bar{x}_v^i$ wäre verletzt.

$\square$

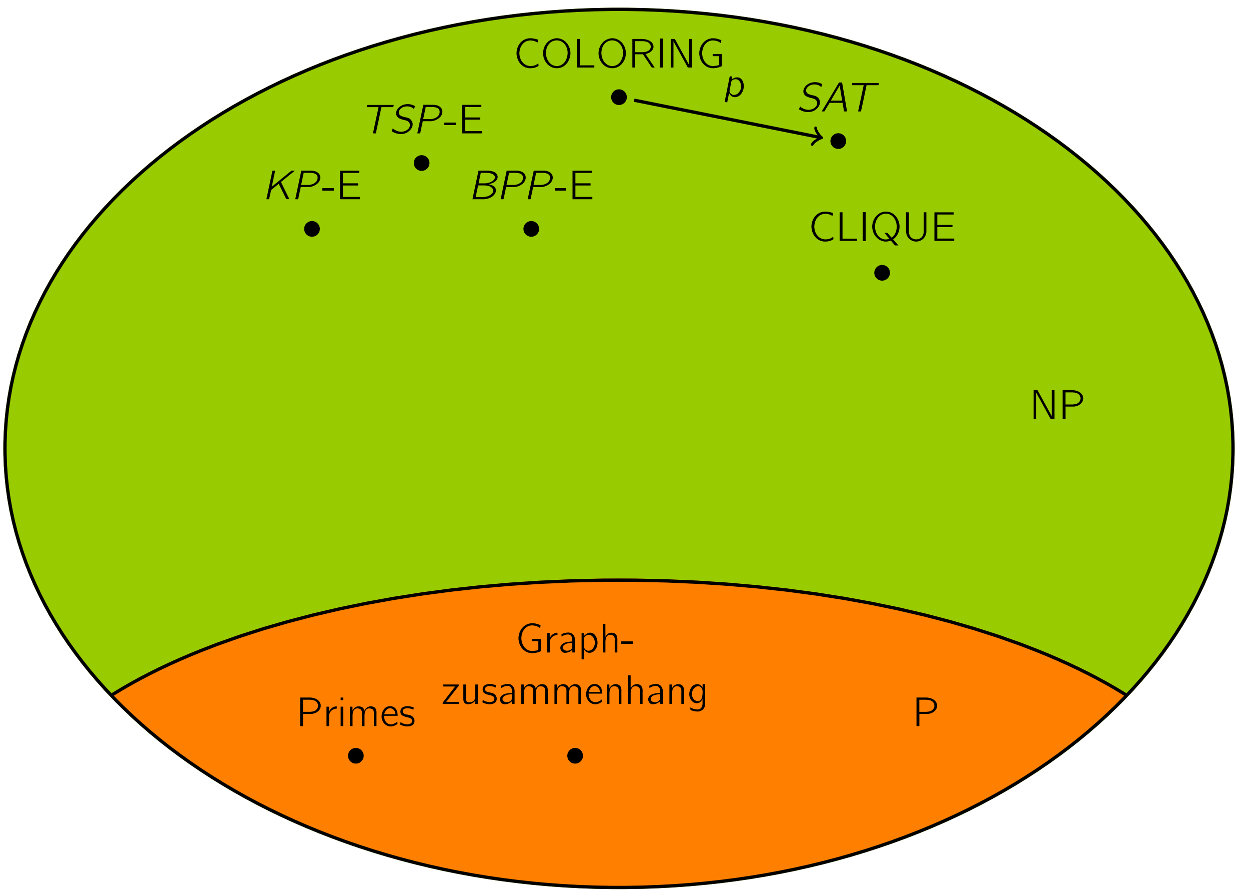
Beispiel einer polyn. Reduktion: COLORING $\leq_p$ SAT

Die Tatsache COLORING $\leq_p$ SAT impliziert das Folgende:

Korollar

Wenn SAT einen Polynomialzeitalgorithmus hat, so hat auch COLORING einen Polynomialzeitalgorithmus.

Die Komplexitätslandschaft



Warnung: Dieser Abbildung liegt die Annahme $P \neq NP$ zu Grunde.

Ausblick

Ein Problem $L \in \mathrm{NP}$ heißt NP-vollständig, wenn für jedes Problem $L' \in \mathrm{NP}$ gilt, dass $L' \leq_p L$.

Satz (Cook und Levin)

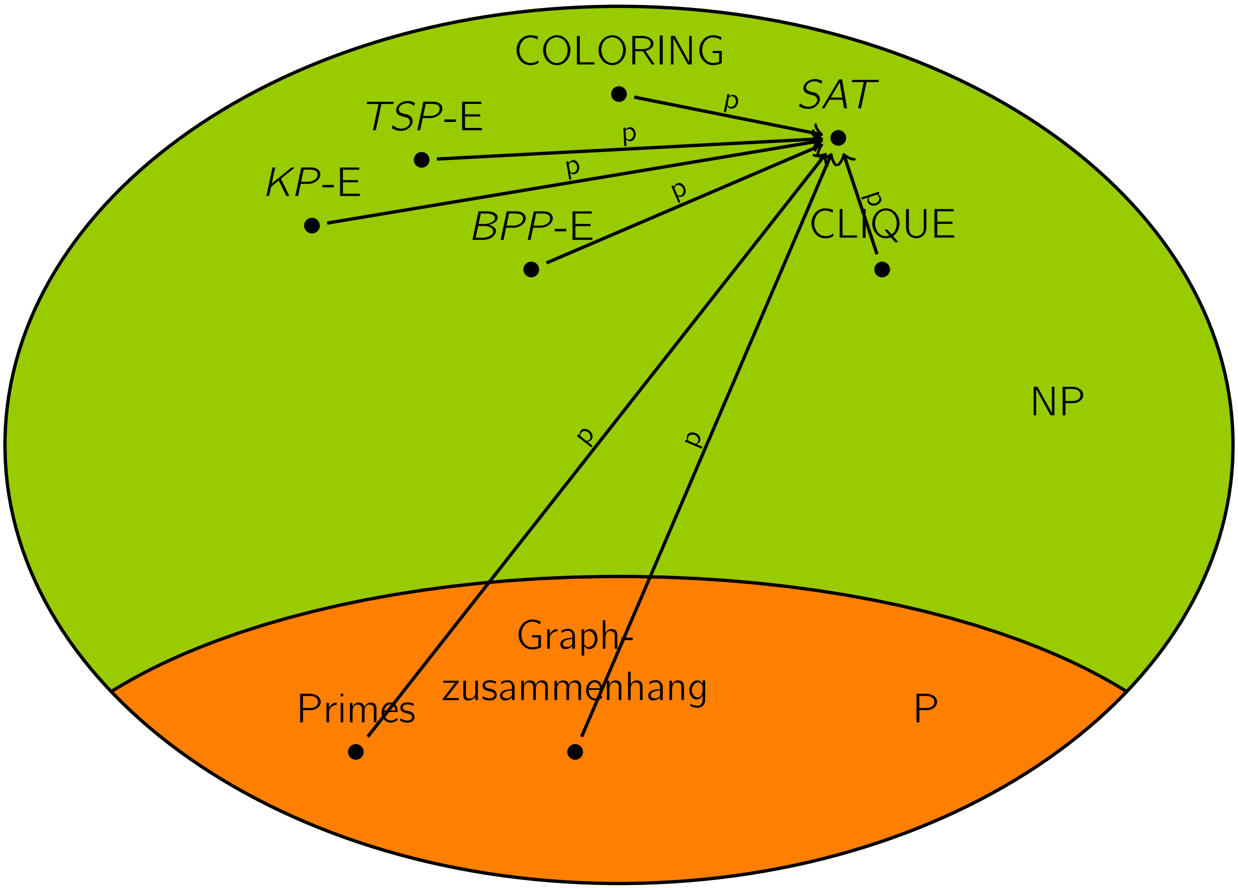
SAT ist NP*-vollständig.*

Es folgt: Wenn SAT einen Polynomialzeitalgorithmus hätte, so gäbe es auch einen Polynomialzeitalgorithmus für jedes andere Problem aus NP und somit wäre $\mathrm{P} = \mathrm{NP}$.

Im Umkehrschluss gilt:

SAT hat keinen Polynomialzeitalgorithmus, außer wenn $\mathrm{P} = \mathrm{NP}$.

Die Komplexitätslandschaft



Warnung: Dieser Abbildung liegt die Annahme $\mathrm{P} \neq \mathrm{NP}$ zu Grunde.