VL-13: Polynomielle Reduktionen

(Berechenbarkeit und Komplexität, WS 2018)

Gerhard Woeginger

WS 2018, RWTH

Organisatorisches

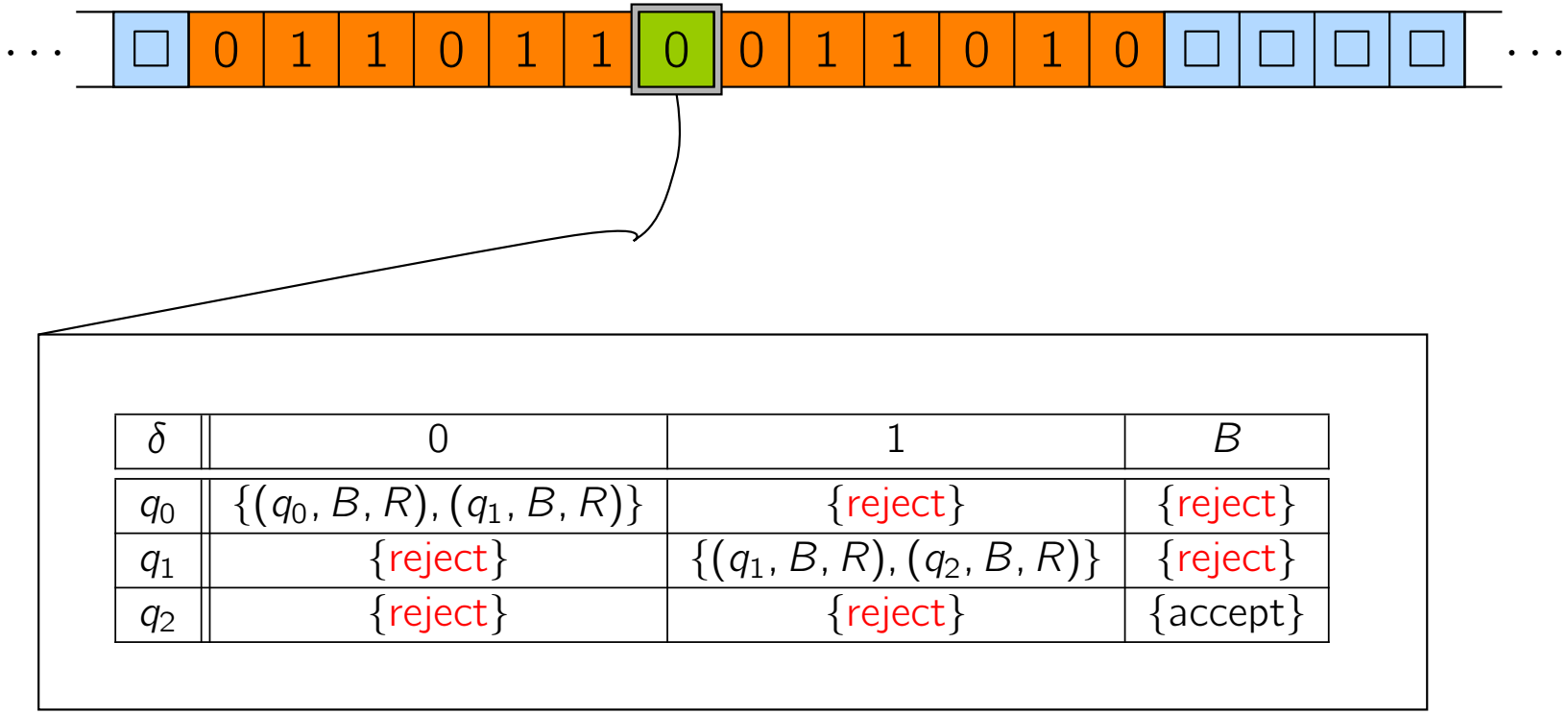
- Nächste Vorlesungen:
  Donnerstag, Dezember 20, 12:30–14:00 Uhr, Aula
  Donnerstag, Januar 11, 12:30–14:00 Uhr, Aula

- **Keine Vorlesung:** Freitag, Dezember 21

- Webseite:
  `http://algo.rwth-aachen.de/Lehre/WS1819/BuK.php`
  ($\longrightarrow$ Arbeitsheft zur Berechenbarkeit)

Wiederholung

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.: P und NP

Komplexitätsklassen P und NP

P ist die Klasse aller Entscheidungsprobleme,
für die es einen polynomiellen Algorithmus gibt.

NP ist die Klasse aller Entscheidungsprobleme,
die durch eine NTM M erkannt werden,
deren Worst Case Laufzeit $t_M(n)$ polynomiell beschränkt ist.

Satz (Zertifikat Charakterisierung von NP)

Eine Sprache $L \subseteq \Sigma^*$ liegt genau dann in NP,
wenn es einen polynomiellen (deterministischen) Algorithmus V
und ein Polynom p mit der folgenden Eigenschaft gibt:

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

Wdh.: Probleme in NP

Problem: Satisfiability (SAT)

Eingabe: Eine Boole'sche Formel φ in CNF über einer Boole'schen Variablenmenge $X = \{x_1, \ldots, x_n\}$

Frage: Existiert eine Wahrheitsbelegung von X, die φ erfüllt?

Problem: CLIQUE

Eingabe: Ein ungerichteter Graph $G = (V, E)$; eine Zahl k

Frage: Enthält G eine Clique mit $\geq k$ Knoten?

Problem: Hamiltonkreis (Ham-Cycle)

Eingabe: Ein ungerichteter Graph $G = (V, E)$

Frage: Besitzt G einen Hamiltonkreis?

Wdh.: Postsches Correspondenzproblem / Übung

Definition (Postsches Correspondenzproblem, PCP)

Eine Instanz des PCP besteht aus einer endlichen Menge

$$K = \left\{ \left[\frac{x_1}{y_1}\right], \ldots, \left[\frac{x_k}{y_k}\right] \right\}.$$

Die Frage ist, ob es eine (nicht-leere) correspondierende Folge $\langle i_1, \ldots, i_n \rangle$ gibt, sodass $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$.

Satz (???)

PCP $\in NP$

Beweis:

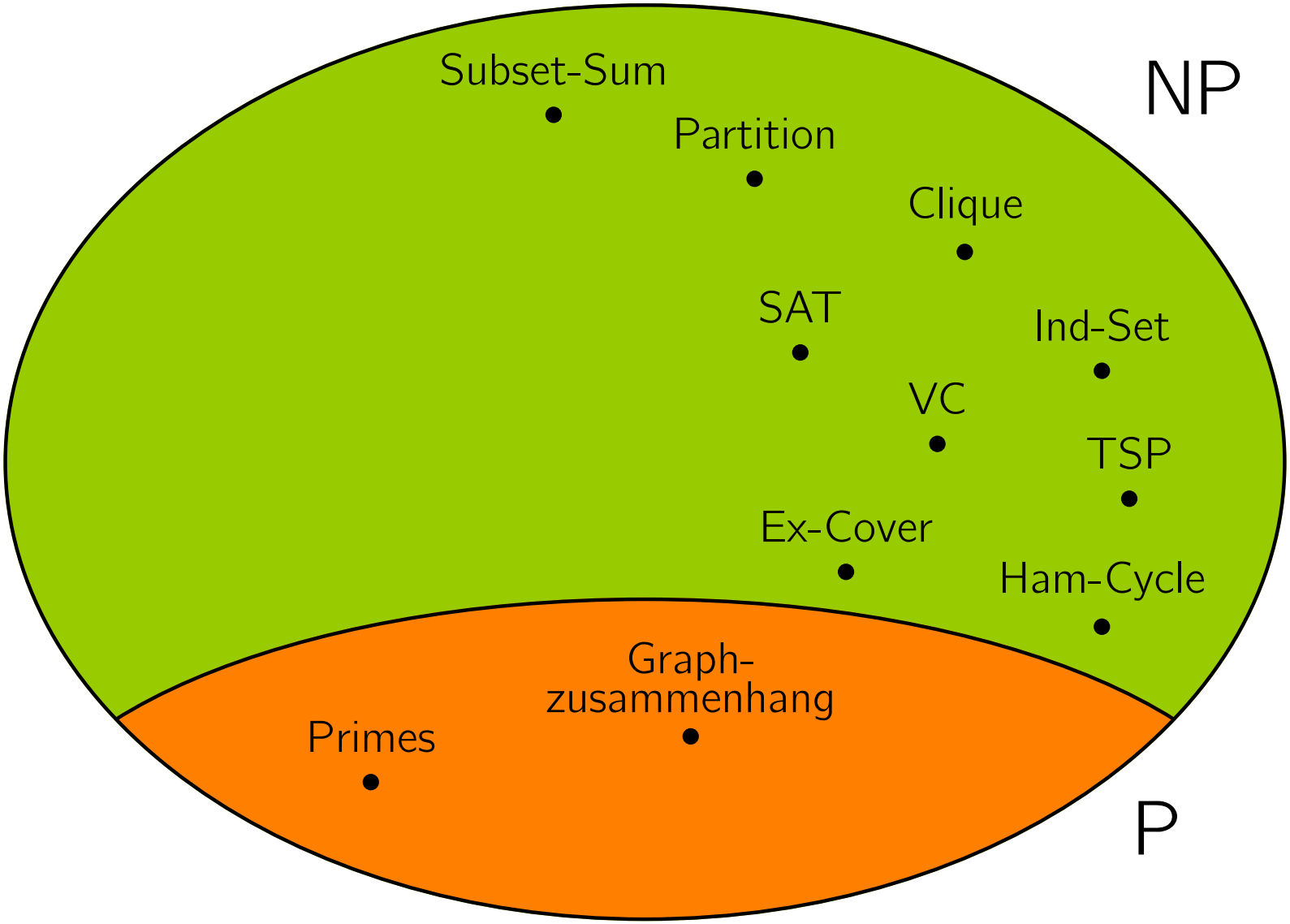
- Zertifikat = correspondierende Folge $\langle i_1, \ldots, i_n \rangle$
- Verifizierer überprüft ob $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

Wdh.: Die grosse offene Frage der Informatik

P=NP ?

Falls die Lösung eines Problems einfach zu überprüfen ist,
ist es dann auch immer einfach, die Lösung zu entdecken?

Wdh.: Die Komplexitätslandschaft



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

Vorlesung VL-13
Polynomielle Reduktionen

- Lösung finden versus Lösbarkeit entscheiden
- Optimieren versus Lösbarkeit entscheiden

- Die Komplexitätsklasse EXPTIME
- Polynomielle Reduktionen

Lösung finden
versus
Lösbarkeit entscheiden

Lösung finden vs Lösbarkeit entscheiden

Ein beliebiges Entscheidungsproblem in NP

Eingabe: Ein diskretes Objekt X.

Frage: Existiert für dieses Objekt X eine Lösung Y?

Dilemma:

- Das Entscheidungsproblem beschäftigt sich nur mit der Frage, ob eine derartige Lösung Y existiert
- Aber eigentlich will man das Lösungsobjekt Y auch genau bestimmen, und dann damit arbeiten

Ausweg:

- Ein schneller Algorithmus für das Entscheidungsproblem liefert (durch wiederholte Anwendung) oft auch einen schnellen Algorithmus zum Berechnen eines expliziten Lösungsobjekts

Beispiel: SAT (1)

Problem: Satisfiability (SAT)

Eingabe: Boole'sche Formel φ in CNF über $X = \{x_1, \ldots, x_n\}$

Frage: Existiert eine Wahrheitsbelegung von X, die φ erfüllt?

- Wenn wir in φ eine Variable $x := 1$ setzen,
  so werden alle Klauseln mit Literal x dadurch erfüllt und
  in allen Klauseln mit Literal $\bar{x}$ fällt dieses Literal einfach weg.
- Wir erhalten wir eine kürzere CNF-Formel $\varphi[x = 1]$.
- Analog erhalten wir mit $x := 0$ die CNF-Formel $\varphi[x = 0]$.

Beispiel

Für $\varphi = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \wedge (\neg y \vee z) \wedge (u \vee z)$

gilt $\varphi[y = 1] = (\neg x \vee \neg z) \wedge (z) \wedge (u \vee z)$

und $\varphi[z = 0] = (x \vee y) \wedge (\neg y) \wedge (u)$

Beispiel: SAT (2)

Wir betrachten SAT Instanzen mit n Variablen und m Klauseln.

Satz

Angenommen, Algorithmus A entscheidet SAT Instanzen in $T(n, m)$ Zeit.
Dann gibt es einen Algorithmus B, der für erfüllbare SAT Instanzen in $n \cdot T(n, m)$ Zeit eine erfüllende Wahrheitsbelegung konstruiert.

Beweis:

- Wir fixieren der Reihe nach die Wahrheitswerte von $x_1, x_2, \ldots, x_n$.
- FOR $i = 1, 2, \ldots, n$ DO
  Wenn $\varphi[x_i = 1]$ erfüllbar, so setze $x_i := 1$ und $\varphi := \varphi[x_i = 1]$
  Andernfalls setze $x_i := 0$ und $\varphi := \varphi[x_i = 0]$
- Am Ende ergeben die fixierten Wahrheitswerte von $x_1, x_2, \ldots, x_n$ eine erfüllende Wahrheitsbelegung für φ

Beispiel: CLIQUE

Problem: CLIQUE

Eingabe: Ein ungerichteter Graph $G = (V, E)$; eine Zahl k

Frage: Enthält G eine Clique mit $\geq k$ Knoten?

- Wenn wir aus G einen Knoten v und alle zu v inzidenten Kanten wegstreichen, so erhalten wir den kleineren Graphen $G - v$
- Falls $G - v$ eine k-Clique enthält, so ist v irrelevant
- Falls $G - v$ aber keine k-Clique enthält, so muss die Nachbarschaft $N[v]$ des Knoten v in $G - v$ eine $(k - 1)$-Clique enthalten

Satz

Angenommen, Algorithmus A entscheidet CLIQUE in $T(n)$ Zeit.
Dann gibt es einen Algorithmus B, der für JA-Instanzen
in $n \cdot T(n)$ Zeit eine k-Clique konstruiert.

Beispiel: Hamiltonkreis

Problem: Hamiltonkreis (Ham-Cycle)

Eingabe: Ein ungerichteter Graph $G = (V, E)$

Frage: Besitzt G einen Hamiltonkreis?

- Wenn wir aus G eine Kante e wegstreichen, so erhalten wir den kleineren Graphen $G - e$
- Falls $G - e$ einen Hamiltonkreis enthält, so ist e irrelevant
- Falls $G - e$ keinen Hamiltonkreis enthält, so ist e nicht irrelevant

Satz

Angenommen, Algorithmus A entscheidet Ham-Cycle in $T(n)$ Zeit.
Dann gibt es einen Algorithmus B, der für JA-Instanzen in $|E| \cdot T(n)$ Zeit einen Hamiltonkreis konstruiert.

Optimieren
versus
Lösbarkeit entscheiden

Optimieren vs Lösbarkeit entscheiden

Definition: Optimierungsproblem

Die Eingabe eines Optimierungsproblems spezifiziert (implizit oder explizit) eine Menge $\mathcal{L}$ von zulässigen Lösungen zusammen mit einer Zielfunktion $f : \mathcal{L} \to \mathbb{N}$, die Kosten, Gewicht, oder Profit misst.

Das Ziel ist es, eine optimale Lösung in $\mathcal{L}$ zu berechnen.
In Minimierungsproblemen sollen die Kosten minimiert werden, und in Maximierungsproblemen soll der Profit maximiert werden.

Dilemma:

> Die Klassen P und NP enthalten keine Optimierungsprobleme, sondern nur Entscheidungsprobleme

Ausweg:

> Optimierungsprobleme können in "sehr ähnliche" Entscheidungsprobleme umformuliert werden

Beispiel: Rucksackproblem

- Beim Rucksackproblem (Knapsack problem, KP) sind n Objekte mit Gewichten $w_1, \ldots, w_n$ und Profiten $p_1, \ldots, p_n$ gegeben
- Ausserdem ist eine Gewichtsschranke b gegeben
- Wir suchen eine Teilmenge K der Objekte, die in einen Rucksack mit Gewichtsschranke b passt und die den Gesamtprofit maximiert

Problem: Rucksack / Knapsack (KP)

Eingabe: Natürliche Zahlen $w_1, \ldots, w_n$, $p_1, \ldots, p_n$, und b

Zulässige Lösung: Menge $K \subseteq \{1, \ldots, n\}$ mit $w(K) := \sum_{i \in K} w_i \leq b$

Ziel: Maximiere $p(K) := \sum_{i \in K} p_i$

Entscheidungsproblem:

- Die Eingabe enthält zusätzlich eine Schranke γ für den Profit
- Frage: Existiert eine zulässige Lösung K mit $p(K) \geq \gamma$?

Beispiel: Bin Packing

Beim Bin Packing sollen n Objekte mit Gewichten $w_1, \ldots, w_n$ auf eine möglichst kleine Anzahl von Kisten mit Gewichtslimit b verteilt werden.

Problem: Bin Packing (BPP)

Eingabe: Natürliche Zahlen b und $w_1, \ldots, w_n \in \{1, \ldots, b\}$

Zulässige Lösung: Zahl $k \in \mathbb{N}$ und Funktion $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$
sodass $\forall i \in \{1, \ldots, k\} : \sum_{j \in f^{-1}(i)} w_j \leq b$

Ziel: Minimiere k (= Anzahl der Kisten)

Entscheidungsproblem:

- Die Eingabe enthält zusätzlich eine Schranke γ
- Frage: Existiert eine zulässige Lösung mit $\leq \gamma$ Kisten?

Beispiel: Travelling Salesman

- Beim Travelling Salesman Problem sind Städte $1, \ldots, n$ gegeben, zusammen mit Distanzen $d(i, j)$ für $1 \leq i \neq j \leq n$
- Gesucht ist eine möglichst kurze Rundreise (Hamiltonkreis; Tour) durch alle Städte

Problem: Travelling Salesman (TSP)

Eingabe: Natürliche Zahlen $d(i, j)$ für $1 \leq i \neq j \leq n$

Zulässige Lösung: Permutation π von $1, \ldots, n$

Ziel: Minimiere $d(\pi) := \sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) + d(\pi(n), \pi(1))$

Entscheidungsproblem:

- Die Eingabe enthält zusätzlich eine Schranke γ
- Frage: Existiert eine zulässige Lösung mit Länge $d(\pi) \leq \gamma$?

Optimieren vs Entscheiden

Für ein Optimierungsproblem mit einer Menge $\mathcal{L}$ von zulässigen Lösungen und einer Gewichtsfunktion $f : \mathcal{L} \to \mathbb{N}$ definieren wir das entsprechende Entscheidungsproblem:

Eingabe: Wie im Optimierungsproblem; plus Schranke $\gamma \in \mathbb{N}$

Frage: Existiert eine zulässige Lösung $x \in \mathcal{L}$
mit $f(x) \geq \gamma$ (für Maximierungsprobleme) respektive
mit $f(x) \leq \gamma$ (für Minimierungsprobleme)?

- Mit Hilfe eines Algorithmus, der das Optimierungsproblem löst, kann man immer das entsprechende Entscheidungsproblem lösen. (Wie?)
- Mit Hilfe eines Algorithmus, der das Entscheidungsproblem löst, kann man den optimalen Zielfunktionswert bestimmen (und oft auch die dazugehörende optimale Lösung finden).

Beispiel: Rucksackproblem (1)

Eingabe: Natürliche Zahlen $w_1, \ldots, w_n$, $p_1, \ldots, p_n$; b; γ
Zulässig: Menge $K \subseteq \{1, \ldots, n\}$ mit $w(K) \leq b$

Optimierung: Berechne K mit maximalem $p(K)$
Entscheidung: Existiert K mit $p(K) \geq \gamma$?

Satz

Wenn das Entscheidungsproblem für KP in polynomieller Zeit lösbar ist, so ist auch das Optimierungsproblem für KP in polynomieller Zeit lösbar.

Beweis: Aus polynomiellem Algorithmus A fürs Entscheidungsproblem

- konstruieren wir zuerst einen polynomiellen Algorithmus B, der den optimalen Zielfunktionswert bestimmt
- und dann einen polynomiellen Algorithmus C, der die optimale zulässige Lösung bestimmt

Beispiel: Rucksackproblem (2)

Algorithmus B für (Phase 1)

Wir führen eine Binäre Suche mit den folgenden Parametern durch:

- Der minimale Profit ist 0.
- Der maximale Profit ist $P := \sum_{i=1}^{n} p_i$.
- Wir finden den optimalen Zielfunktionswert durch Binäre Suche über dem Wertebereich $\{0, \ldots, P\}$.
- In jeder Iteration verwenden wir den polynomiellen Algorithmus A (für das Entscheidungsproblem), der uns sagt in welcher Richtung wir weitersuchen müssen

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

Beispiel: Rucksackproblem (3)

Untersuchung der Eingabelänge:

- Die Kodierungslänge einer Zahl $a \in \mathbb{N}$ ist $\kappa(a) := \lceil \log(a+1) \rceil$.
- Die Logarithmusfunktion ist sub-additiv:
  Für alle $a, b \in \mathbb{N}$ gilt $\kappa(a+b) \leq \kappa(a) + \kappa(b)$.
- Die Eingabelänge L des Rucksackproblems beträgt mindestens

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

- Algorithmus B besteht aus $\lceil \log(P+1) \rceil \leq L$ Aufrufen des polynomiellen Algorithmus A
- Also ist die Gesamtlaufzeit von Algorithmus B polynomiell in der Eingabelänge des Rucksackproblems beschränkt

Beispiel: Rucksackproblem (4)

Aus Algorithmus B konstruieren wir nun noch den Algorithmus C, der die optimale zulässige Lösung bestimmt:

Algorithmus C

```
1  K := {1, ..., n};
2  opt := B(K);
3  FOR i := 1 TO n do
4        IF B(K \ {i}) = opt THEN K := K \ {i}; ENDIF;
5  ENDFOR;
6  OUTPUT  K
```

- Algorithmus C besteht im wesentlichen aus $n+1$ Aufrufen des polynomiellen Algorithmus B
- Also ist die Gesamtlaufzeit von Algorithmus C polynomiell in der Eingabelänge des Rucksackproblems beschränkt

Die Komplexitätsklasse EXPTIME

EXPTIME (1): Definition

Definition: Komplexitätsklasse EXPTIME

EXPTIME ist die Klasse aller Entscheidungsprobleme,
die durch eine DTM M entschieden werden, deren Worst Case Laufzeit durch $2^{q(n)}$ mit einem Polynom q beschränkt ist,

Laufzeit-Beispiele: $2^{\sqrt{n}}$, 2^n, 3^n, $n!$, n^n. Aber nicht: 2^{2^n}

Wie verhalten sich die Klassen P und NP zu EXPTIME?

EXPTIME (2): NP $\subseteq$ EXPTIME

Satz

NP $\subseteq$ EXPTIME

- Es sei $L \in NP$
- Dann gibt es ein Polynom p und einen polynomiellen Algorithmus V
  $x \in L \iff \exists y \in \{0,1\}^*,\ |y| \leq p(|x|) :\ V$ akzeptiert $y\#x$

- Wir enumerieren alle Kandidaten $y \in \{0,1\}^*$ mit $|y| \leq p(|x|)$
  Wir testen jeden Kandidaten mit dem Verifizierer V
  Wir akzeptieren, falls V einen der Kandidaten akzeptiert

- Anzahl der Kandidaten $\approx\ 2^{p(|x|)}$
  Zeit pro Kandidat $\approx$ polynomiell in $|x|$ plus $|y|$
  $\Rightarrow$ Gesamtzeit $\approx\ \text{poly}(|x|) \cdot 2^{p(|x|)}$

EXPTIME (3): Zwei Beispiele

Problem: Satisfiability (SAT)

Eingabe: Eine Boole'sche Formel φ in CNF über der Boole'schen Variablenmenge $X = \{x_1, \ldots, x_n\}$

Frage: Existiert eine Wahrheitsbelegung von X, die φ erfüllt?

Problem: Hamiltonkreis (HAM-CYCLE)

Eingabe: Ein ungerichteter Graph $G = (V, E)$ mit $|V| = n$

Frage: Besitzt G einen Hamiltonkreis?

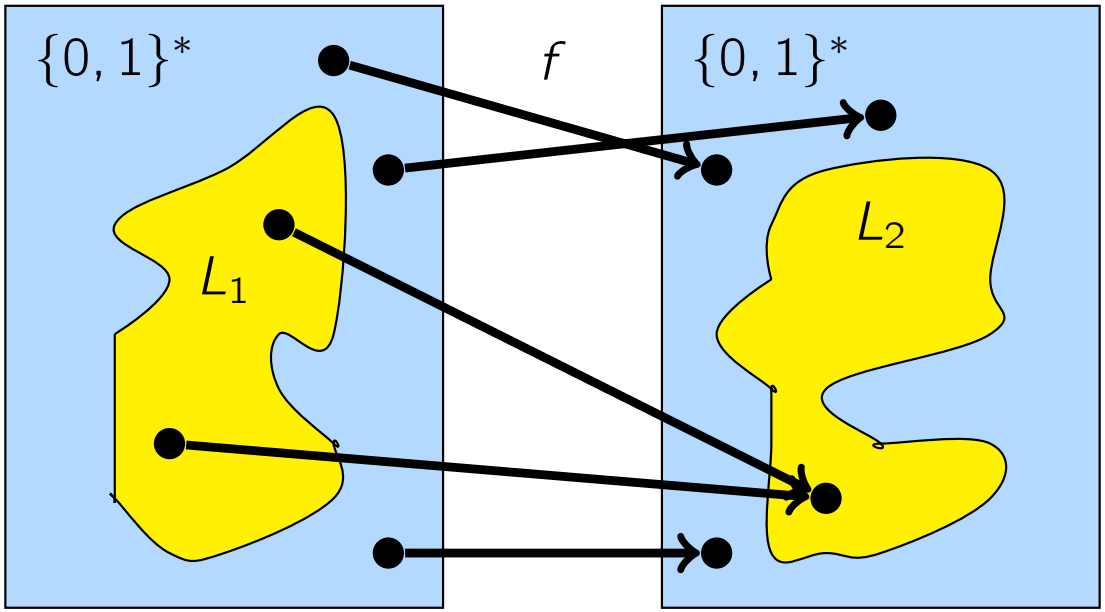
Frage:
Welche (exponentielle) Zeitkomplexität ergibt sich aus dem vorangehenden Beweis für die Probleme SAT und HAM-CYCLE?

Polynomielle Reduktionen

Zur Erinnerung: Eine alte Seite aus Vorlesung VL-06

Definition

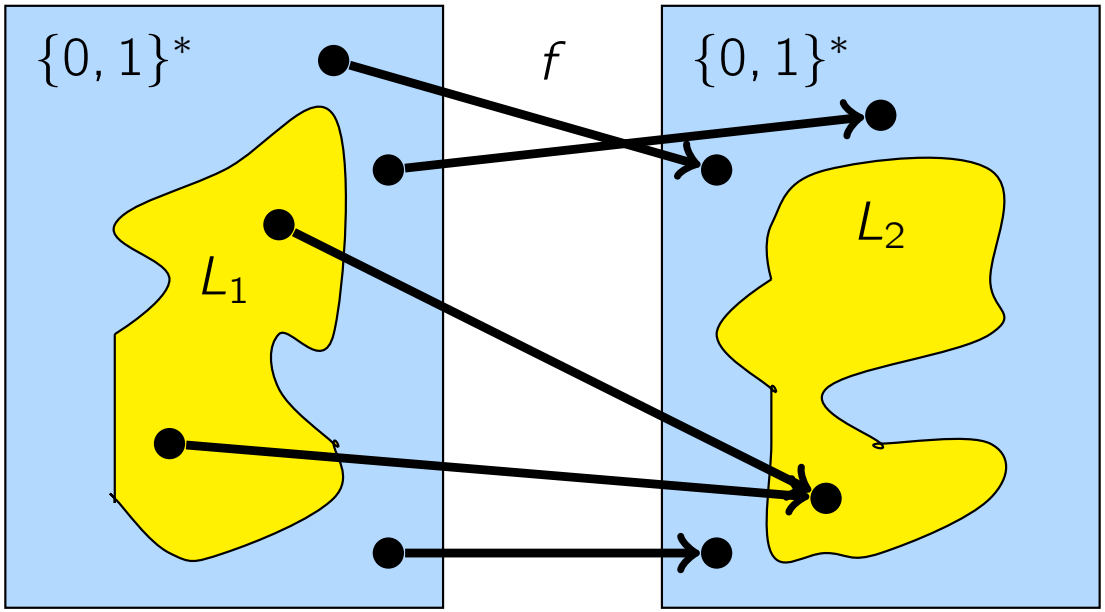
Es seien L_1 und L_2 zwei Sprachen über einem Alphabet Σ.
Dann ist L_1 auf L_2 reduzierbar (mit der Notation $L_1 \leq L_2$),
wenn eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ existiert,
so dass für alle $x \in \Sigma^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.



Polynomielle Reduktionen (1)

Definition

Es seien L_1 und L_2 zwei Sprachen über einem Alphabet Σ.
Dann ist L_1 polynomiell reduzierbar auf L_2 (mit der Notation $L_1 \leq_p L_2$),
wenn eine polynomiell berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ existiert,
so dass für alle $x \in \Sigma^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.



Polynomielle Reduktionen (2)

Satz

Falls $L_1 \leq_p L_2$ und falls $L_2 \in P$, so gilt $L_1 \in P$.

Beweis:

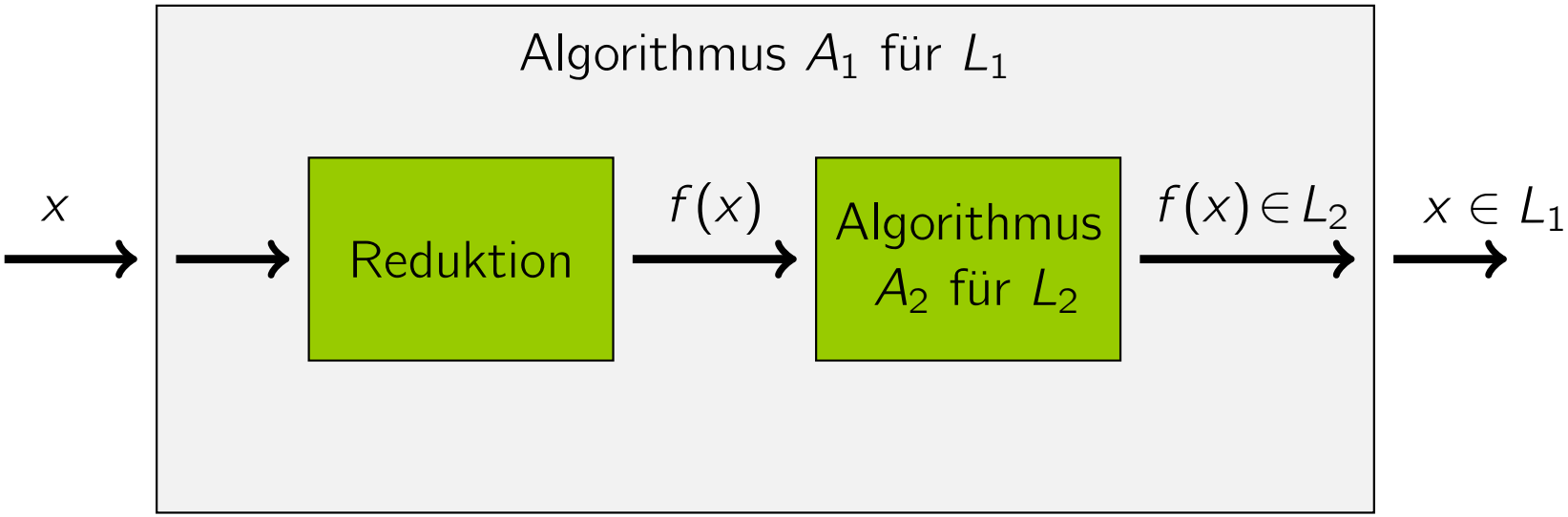
- Die Reduktion f hat die polynomielle Laufzeitschranke $p(\cdot)$
- Der Algorithmus A_2 entscheidet L_2 mit einer polynomiellen Laufzeitschranke $q(\cdot)$

Wir konstruieren einen Algorithmus A_1, der L_1 entscheidet:

Schritt 1: Berechne $f(x)$
Schritt 2: Simuliere Algorithmus A_2 auf $f(x)$
Schritt 3: Akzeptiere x, genau dann wenn A_2 akzeptiert

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

Polynomielle Reduktionen (3)



Beispiel zu Reduktionen: COLORING $\leq_p$ SAT

COLORING $\leq_p$ SAT

Problem: Knotenfärbung / COLORING

Eingabe: Ein ungerichteter Graph $G = (V, E)$; eine Zahl $k \in \mathbb{N}$

Frage: Gibt es eine Färbung $c : V \to \{1, \ldots, k\}$ der Knoten mit k Farben, sodass benachbarte Knoten verschiedene Farben erhalten? $\forall e = \{u, v\} \in E : \; c(u) \neq c(v)$?

Problem: Satisfiability (SAT)

Eingabe: Boole'sche Formel φ in CNF über der Variablenmenge X

Frage: Existiert eine Wahrheitsbelegung von X, die φ erfüllt?

Satz

COLORING $\leq_p$ SAT

COLORING $\leq_p$ SAT: Die Reduktion

Die Boole'schen Variablen

Für jeden Knoten $v \in V$ und für jede Farbe $i \in \{1, \ldots, k\}$
führen wir eine Boole'sche Variable x_v^i ein.

Die Klauseln

Für jeden Knoten $v \in V$
verwenden wir die Klausel $(x_v^1 + x_v^2 + \ldots + x_v^k)$

Für jede Kante $\{u, v\} \in E$ und jede Farbe $i \in \{1, \ldots, k\}$
verwenden wir die Klausel $(\bar{x}_u^i + \bar{x}_v^i)$

- Anzahl der Variablen $= k\,|V|$
- Anzahl der Klauseln $= |V| + k\,|E|$
- Gesamtlänge der Formel $= k\,|V| + 2k\,|E| \;\in\; O(k|V|^2)$

COLORING $\leq_p$ SAT: Korrektheit (1)

Graph G hat k-Färbung $\Rightarrow$ Formel φ ist erfüllbar

- Es sei c eine k-Färbung für G
- Für jeden Knoten v mit $c(v) = i$ setzen wir $x_v^i = 1$.
  Alle anderen Variablen setzen wir auf 0.
- Für jeden Knoten $v \in V$ ist $(x_v^1 + x_v^2 + \ldots + x_v^k)$ erfüllt
- Für $\{u, v\} \in E$ und $i \in \{1, \ldots, k\}$ ist $(\bar{x}_u^i + \bar{x}_v^i)$ erfüllt
  (Andernfalls hätten beide Knoten u und v die selbe Farbe i.)
- Ergo: Diese Wahrheitsbelegung erfüllt die Formel φ

COLORING $\leq_p$ SAT: Korrektheit (2)

Formel φ ist erfüllbar $\Rightarrow$ Graph G hat k-Färbung

- Wir betrachten eine beliebige erfüllende Belegung für φ
- Wegen der Klausel $(x_v^1 + x_v^2 + \ldots + x_v^k)$ gibt es für jeden Knoten v mindestens eine Farbe i mit $x_v^i = 1$
- Für jeden Knoten wählen wir eine beliebige derartige Farbe aus
- Wir behaupten: $c(u) \neq c(v)$ gilt für jede Kante $\{u, v\} \in E$
- Beweis: Falls $c(u) = c(v) = i$, dann gilt $x_u^i = x_v^i = 1$. Dann wäre aber die Klausel $(\bar{x}_u^i + \bar{x}_v^i)$ verletzt

COLORING $\leq_p$ SAT: Konsequenzen

Aus unserer Reduktion COLORING $\leq_p$ SAT folgt:

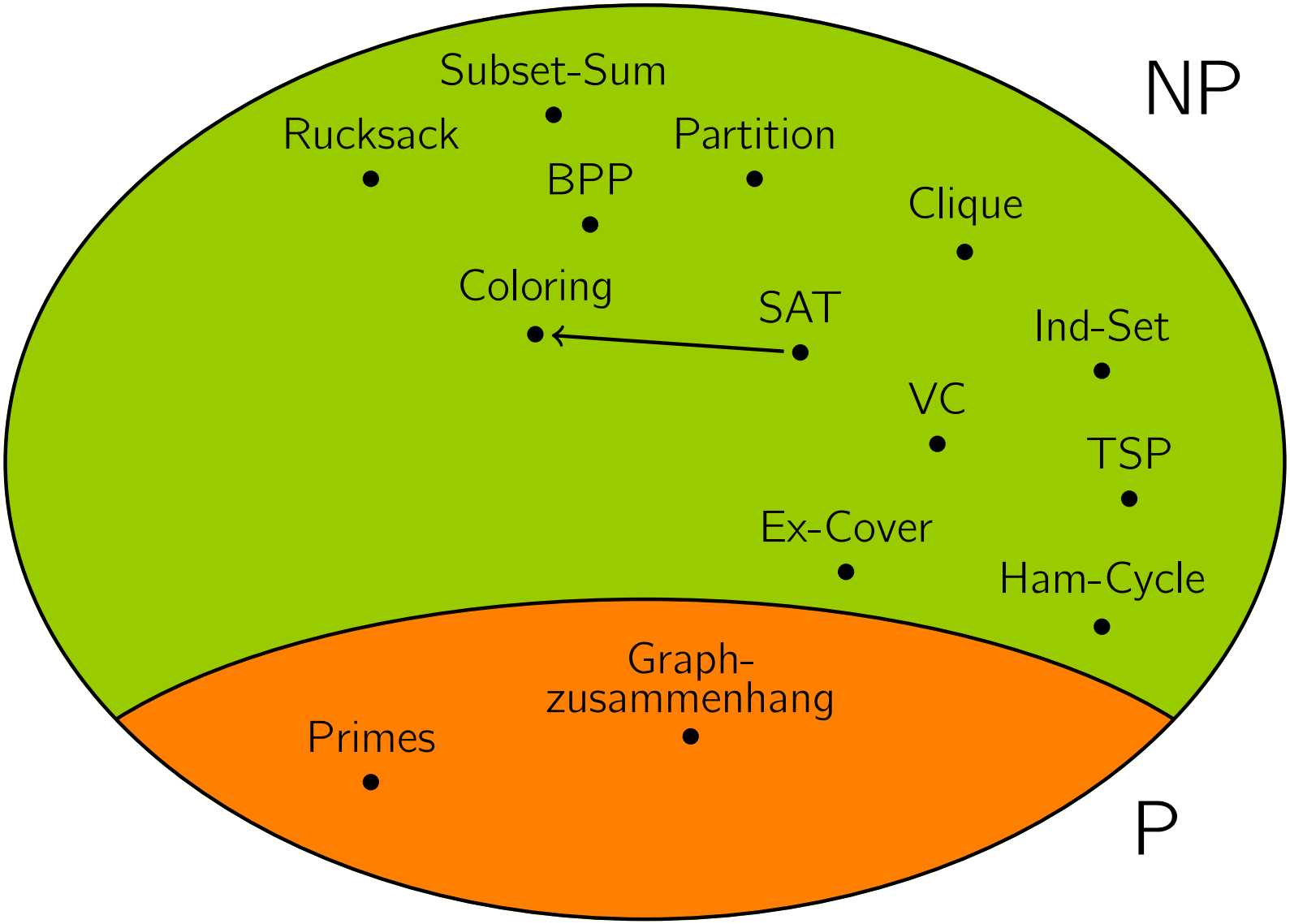
Folgerung

Wenn SAT einen polynomiellen Algorithmus hat,
so hat auch COLORING einen polynomiellen Algorithmus.

Folgerung (logisch äquivalent)

Wenn COLORING keinen polynomiellen Algorithmus hat,
so hat auch SAT keinen polynomiellen Algorithmus.

Die Komplexitätslandschaft



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

Übung: Ex-Cover $\leq_p$ SAT

Problem: Exact Cover (Ex-Cover)

Eingabe: Eine endliche Menge X; Teilmengen $S_1, \ldots, S_m$ von X

Frage: Existiert eine Indexmenge $I \subseteq \{1, \ldots, m\}$,
sodass die Mengen S_i mit $i \in I$ eine Partition von X bilden?

Übung

Zeigen Sie: Ex-Cover $\leq_p$ SAT

Nicht-Beispiel zu Reduktionen: Vertex Cover $\leq_p$ SAT

Vertex Cover $\leq_p$ SAT

Problem: Vertex Cover (VC)

Eingabe: Ein ungerichteter Graph $G = (V, E)$; eine Zahl $k \in \mathbb{N}$

Frage: Enthält G ein Vertex Cover mit $\leq k$ Knoten?

Vertex Cover $S \subseteq V$ enthält mindestens einen Endpunkt von jeder Kante

Problem: Satisfiability (SAT)

Eingabe: Boole'sche Formel φ in CNF über der Variablenmenge X

Frage: Existiert eine Wahrheitsbelegung von X, die φ erfüllt?

Satz (???)

Vertex Cover $\leq_p$ SAT

Vertex Cover $\leq_p$ SAT: Die Reduktion (???)

Die Boole'schen Variablen

Für jeden Knoten $v \in V$
führen wir eine Boole'sche Variable x_v ein.

Die Klauseln

Für jede Kante $\{u, v\} \in E$
verwenden wir die Klausel $(x_u + x_v)$

Für jede $(k+1)$-elementige Teilmenge $S \subseteq V$
verwenden wir die Klausel $\bigvee_{v \in S} \bar{x}_v$

- Anzahl der Variablen $= |V|$
- Anzahl der Klauseln $\approx |V|^k$
- Gesamtlänge der Formel $\approx k\,|V|^k$ $\leftarrow$!!!#!!&!!!!!!!