Kapitel 2: NP und NP-Vollständigkeit

<u>Idee:</u> P: die Klasse aller effizient lösbaren Probleme

NP: die Klasse aller Probleme, bei denen Lösungen effizient verifiziert werden können

<u>Definition 2.1</u> (Die Klasse NP)

Eine Sprache $L \in \{0,1\}^*$ liegt in der Klasse NP, falls es ein Polynom $p: \mathbb{N} \to \mathbb{N}$ und eine Polynomialzeit-TM M gibt, s.d. f.a. $x \in \{0,1\}^*$ gilt:

$$x \in L \iff \text{ex. } u \in \{0,1\}^{p(|x|)} \text{ s.d. } M(\langle x, u \rangle) = 1.$$

Die TM M wird <u>Verifizierer</u> für L genannt.

Falls $M(\langle x,u \rangle) = 1$ für $x \in L$ und $u \in \{0,1\}^{p(|x|)}$ gilt, so heißt u ein <u>Zertifikat</u> (oder <u>Zeuge</u>) für x (bzgl. L und M).

<u>Fakt 2.2</u>: $P \subseteq NP$

<u>Beweis:</u> Sei $L \in P$. Dann gibt es eine Polynomialzeit-TM M, die L entscheidet. Wähle $p: \mathbb{N} \to \mathbb{N}$ mit $p(n) = 0$ f.a. $n \in \mathbb{N}$ (kurz: $p \equiv 0$). Damit ist M ein Verifizierer für L, und somit $L \in NP$. $\square$

Beispiel 2.3: INDSET $\in$ NP

Zur Erinnerung

$$\text{INDSET} = \{ \langle G, k \rangle : \text{G ist ein ungerichteter Graph, der eine unabhängige Menge der Größe k besitzt} \}$$

Beweisidee:

Unser Verifizierer M für INDSET bekommt neben G und k noch eine Liste u aus k Knoten von G als Eingabe. M testet dann, ob u eine Liste aus k unabhängigen Knoten ist (d.h. ob es in G keine Kante zwischen zwei Knoten der Liste gibt).

Beachte: Länge der Eingabe $\llcorner\langle G, k\rangle\lrcorner$: $O(n^2 + \log k)$ Bits, wobei $n = |V(G)|$.

Länge der Kodierung der Liste u: $O(k \cdot \log n)$ Bits.

Dies stellt aber kein Problem dar, da wir nur solche k berücksichtigen müssen, die $\leq n$ sind (für $k > n = |V(G)|$ gibt es keine unabhängige Menge der Größe k).

$\square$

P, NP und Exponentialzeit

Definition 2.4 (Die Klasse EXP)

$$\mathrm{EXP} := \bigcup_{c \geq 1} \mathrm{DTIME}\left(2^{(n^c)}\right)$$

Fakt 2.5: $P \subseteq NP \subseteq EXP$

Beweis: $P \subseteq NP$ gilt gemäß Fakt 2.2.

Zu $NP \subseteq EXP$: Sei $L \in NP$. Sei M ein Verifizierer für L und gebe das Polynom $p: \mathbb{N} \to \mathbb{N}$ die Länge der Zertifikate an.

Eine TM M', die L entscheidet, kann bei Eingabe $x \in \{0,1\}^*$ wie folgt vorgehen

Phase 1: Erzeuge auf einem Arbeitsband eine Liste aller $2^{p(|x|)}$ Bitstrings der Länge $\{0,1\}^{p(|x|)}$

Phase 2: Lass nacheinander für jeden dieser Bitstrings u die TM M bei Eingabe $\langle x, u \rangle$ laufen. Falls $M(\langle x, u \rangle) = 1$, so halte an mit Ausgabe 1

Phase 3: Halte an mit Ausgabe 0 (da in Phase 2 kein Zeuge gefunden wurde)

Laufzeit von M': $$2^{O(p(|x|))} \cdot \Big(\underbrace{1}_{\text{für Phase 1}} + \underbrace{\mathrm{poly}(|x| + p(|x|))}_{\text{für Phase 2}} \Big)$$

$$\leq 2^{c \cdot |x|}$$, für eine geeignete Konstante c

Somit: $L \in EXP$. $\square$

Nichtdeterministische Turingmaschinen

Eine nichtdeterministische Turingmaschine (kurz: NDTM) ist analog zur TM definiert (vgl Definition 1.1), mit den folgenden Unterschieden:

- an Stelle einer Transitionsfunktion δ gibt es zwei Transitionsfunktionen: δ_0 und δ_1
- zusätzlich zum Haltezustand q_{halt} gibt es einen speziellen Zustand q_{accept}.

Während eines Laufs kann M in jedem Schritt auswählen, ob sie diesen Schritt gemäß δ_0 oder gemäß δ_1 durchführt — dies sind die "nichtdeterministischen Entscheidungen", die M während eines Laufs treffen kann.

Für ein Eingabewort $x \in \{0,1\}^*$ sagen wir:

- M akzeptiert x (kurz: $M(x) = 1$), falls es bei Eingabe x eine Folge nichtdeterministischer Entscheidungen gibt, die M in den Zustand q_{accept} überführt
- M verwirft x (kurz: $M(x) = 0$), falls bei Eingabe x für jede Folge nichtdeterministischer Entscheidungen gilt: M hält an, ohne irgendwann im Zustand q_{accept} gewesen zu sein.

Wir sagen: <u>M läuft in Zeit $T(n)$</u>,

falls f.a. $x \in \{0,1\}^*$ und für jede Wahl nichtdeterministischer Entscheidungen gilt: Nach $\leq T(|x|+1)$ Schritten erreicht M bei Eingabe x einen der Zustände q_{halt}, q_{accept}.

<u>Definition 2.6</u> (NTIME($T(n)$))

Sei $T: \mathbb{N} \to \mathbb{N}$.

NTIME($T(n)$) := $\{$ $L \subseteq \{0,1\}^*$: Es gibt eine NDTM M, die in Zeit $O(T(n))$ läuft, und f.a. $x \in \{0,1\}^*$ gilt: $x \in L \iff M(x) = 1$ $\}$.

Wir sagen: M <u>entscheidet</u> L in Zeit $O(T(n))$

<u>Theorem 2.7</u>

$$NP = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c).$$

(NP ist also die Klasse aller <u>nicht</u>deterministisch in <u>P</u>olynomialzeit lösbaren Probleme.)

<u>Beweis:</u>

<u>Idee:</u> Ein Zertifikat $u \in \{0,1\}^*$ gemäß Def. 2.1 entspricht bzw. repräsentiert eine Folge nichtdeterministischer Entscheidungen.

<u>Genauer:</u>

"$\supseteq$": Sei N eine NDTM, die eine Sprache L in Zeit n^c entscheidet, für ein $c \in \mathbb{N}$.

Für jedes $x \in L$ gibt es eine Folge nichtdeterministischer Entscheidungen, so dass N bei Eingabe x den Zustand q_{accept} erreicht. Diese Folge kann durch einen Bitstring u der Länge n^c repräsentiert werden.

Klar: Es gibt eine (det.) TM M, die bei Eingabe $\langle x, u \rangle$ genau dasselbe tut wie N bei Eingabe x und den gemäß u beschriebenen nichtdeterministischen Entscheidungen.

Somit: $L \in NP$ gemäß Def 2.1.

"$\subseteq$": Sei $L \in NP$, M ein Verifizierer für L. Sei $p: \mathbb{N} \to \mathbb{N}$ ein Polynom, das die Länge der Zertifikate angibt, und sei n^c die Laufzeit von M.

Eine NDTM N, die L in Polynomialzeit entscheidet, kann bei Eingabe $x \in \{0,1\}^*$ wie folgt vorgehen:

Phase (1): Nutze die nichtdet. Entscheidungen, um einen beliebigen Bitstring u der Länge $p(|x|)$ auf ein Arbeitsband zu schreiben.

Phase (2): Simuliere M bei Eingabe $\langle x, u \rangle$.

Klar: N entscheidet L in Polynomialzeit.

Somit: $L \in \bigcup_{c \in \mathbb{N}} NTIME(n^c)$

$\square$

2.2 Reduzierbarkeit und NP-Vollständigkeit

Definition 2.8 (Polynomialzeit-Reduktion)

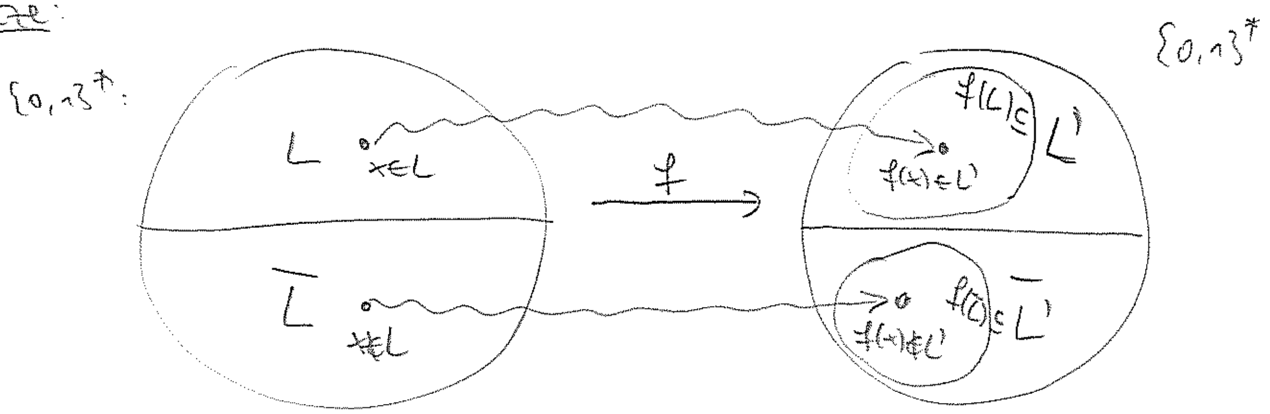
Seien $L, L' \subseteq \{0,1\}^*$.

L ist <u>Polynomialzeit-reduzierbar</u> auf L', kurz: <u>$L \leq_p L'$</u>,

falls es eine in Polynomialzeit berechenbare Funktion $f: \{0,1\}^* \to \{0,1\}^*$ gibt, s.d. f.a. $x \in \{0,1\}^*$ gilt:

$$x \in L \iff f(x) \in L'.$$

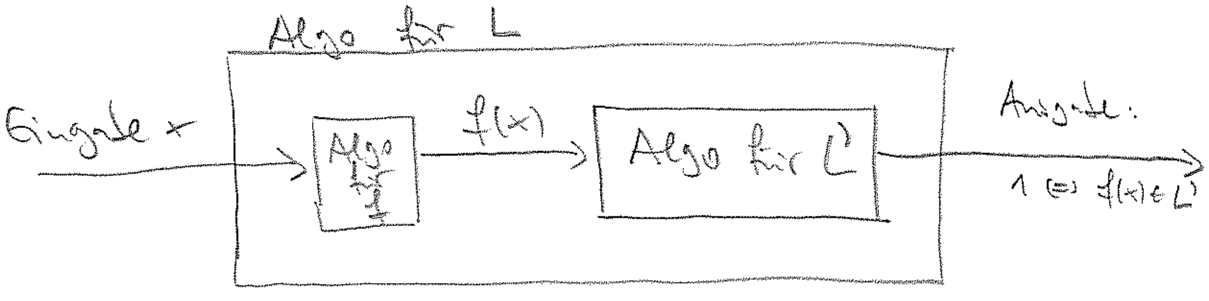
Skizze:



Klar:

Falls $L \leq_p L'$, so ist L bzgl. "Polynomialzeit-Algorithmen" höchstens so schwer wie L'.

Denn: Ein Algo. für L' kann mittels f wie folgt zum Lösen von L benutzt werden:



Bemerkung:

Polynomialzeit-Reduktionen werden auch
Polynomialzeit-Karp-Reduktionen oder
(Polynomialzeit) many-to-one-Reduktionen genannt

Fakt 2.9

Die Relation $\leq_p$ ist transitiv, d.h.:

Falls $L \leq_p L'$ und $L' \leq_p L''$, so $L \leq_p L''$

Beweis: Seien f und f' Polynomialzeit-Reduktionen von L auf L' bzw. von L' auf L''.
Man sieht leicht, dass dann $f'': \{0,1\}^* \to \{0,1\}^*$ mit

$$f''(x) := f'(f(x)) \quad (\forall\, x \in \{0,1\}^*)$$

eine Polynomialzeit-Reduktion von L auf L'' ist.

Definition 2.10 (NP-Härte und NP-Vollständigkeit)

(a) $L' \subseteq \{0,1\}^*$ heißt NP-hart, falls für alle $L \in NP$ gilt: $L \leq_p L'$.

(b) $L' \subseteq \{0,1\}^*$ heißt NP-vollständig, falls $L' \in NP$ und L' NP-hart ist

Fakt 2.11 Sei $L \subseteq \{0,1\}^*$

(a) $(L$ NP-hart und $L \in P) \Rightarrow P = NP$

(b) Falls L NP-vollständig ist, so gilt:

$$L \in P \Leftrightarrow P = NP$$

Beweis: Übung!

Hier ein Beispiel für eine (recht unnatürliche) Sprache, deren NP-Vollständigkeit leicht nachzuweisen is

Theorem 2.12

Die folgende Sprache ist NP-vollständig:

$$\mathrm{TMSAT} := \{ \langle \alpha, x, 1^n, 1^t \rangle : \text{ ex } u \in \{0,1\}^n \text{ s.d. } M_\alpha \text{ bei Eingabe } \langle x, u \rangle \text{ nach } \le t \text{ Schritten den Wert 1 ausgibt} \}$$

Beweis:

TMSAT $\in$ NP: Rate u und nutze dann die universelle TM $\mathcal{U}$, um die ersten t Schritte von M_α bei Eingabe $\langle x, u \rangle$ zu simulieren. (vgl. Theorem 1.14 und Bemerkung 1.

Details: Übung!

NP-Härte von TMSAT:

Sei $L \in NP$ beliebig. Zu zeigen: $L \leq_p$ TMSAT.

Dazu sei gemäß Def 2.1 M ein Verifizierer für L, und sei $p: \mathbb{N} \to \mathbb{N}$ ein Polynom, das die Länge der Zertifikate angibt. D.h. es gilt f.a. $x \in \{0,1\}^*$:

$$x \in L \iff \text{ex } u \in \{0,1\}^{p(|x|)} \text{ s.d. } M(\langle x, u \rangle) = 1.$$

Sei $q: \mathbb{N} \to \mathbb{N}$ ein Polynom, das die Laufzeit von M angibt.

Unsere Polynomialzeit-Reduktion f von L auf TMSAT bildet jedes Wort $x \in \{0,1\}^*$ ab auf

$$f(x) := \langle \llcorner M \lrcorner, x, 1^{p(|x|)}, 1^{q(|x|+p(|x|))} \rangle$$

(klar: Das geht tatsächlich in Polynomialzeit, denn $\llcorner M \lrcorner$ ist fest, x ist die Eingabe und p, q sind Polynome).

Außerdem gilt:

$\langle \llcorner M \lrcorner, x, 1^{p(|x|)}, 1^{q(|x|+p(|x|))} \rangle \in$ TMSAT

$\underset{\text{Def TMSAT}}{\iff}$ ex. $u \in \{0,1\}^{p(|x|)}$ s.d. M bei Eingabe $\langle x, u \rangle$ nach $\leq q(|x|+p(|x|))$ Schritten 1 ausgibt

$\underset{\text{Wahl von } M}{\iff}$ $x \in L$.

Somit gilt: $L \leq_p$ TMSAT.

$\square$

2.3 Eine Sammlung NP-vollständiger Probleme

Probleme, die aussagenlogische Formeln betreffen:

Das aussagenlogische Erfüllbarkeitsproblem SAT

Eingabe: Eine aussagenlogische Formel φ

Frage: Ist φ erfüllbar?

Notation:

- Ist φ eine aussagenlogische Formel über den aussagenlogischen Variablen $u_1, \dots, u_n$, und ist $z \in \{0,1\}^n$, so schreiben wir $\varphi(z)$, um den Wahrheitswert von φ unter der Belegung zu bezeichnen, die jede Variable u_i mit dem Wahrheitswert z_i belegt (für $1 \le i \le n$).
- φ heißt erfüllbar, falls es ein $z \in \{0,1\}^n$ gibt mit $\varphi(z) = 1$.
- Eine (disjunktive) Klausel (der Länge k) ist eine Formel der Form $(\ell_1 \vee \dots \vee \ell_k)$, wobei jedes ℓ_j eine Variable oder eine negierte Variable ist (die ℓ_j heißen Literale)
- Eine Formel ist in konjunktiver Normalform (kurz: CNF), falls sie eine Konjunktion von Klauseln ist

- Eine Formel ist in kCNF (für $k \in \mathbb{N}$), falls sie in CNF ist und jede Klausel die Länge $\leq k$ hat.

Sei $k \in \mathbb{N}$. Das Problem k-SAT:

Eingabe: Eine kCNF-Formel φ

Frage: Ist φ erfüllbar?

Die ersten beiden natürlichen Probleme, deren NP-Vollständigkeit nachgewiesen wurde, waren SAT und 3SAT:

Theorem 2.13 (Der Satz von Cook und Levin, 1971)

(a) SAT ist NP-vollständig

(b) 3SAT ist NP-vollständig

Beweis: ... ist aus der Veranstaltung "Algorithmentheorie" bekannt.

(Wiederholung: Übung!)

$\square$

Weiterhin ist aus "Algorithmentheorie" bekannt, dass 2SAT in P liegt.

Hausaufgabe: Lesen Sie Kapitel 6 des Skripts zur Veranstaltung "Algorithmentheorie".

Aus der Veranstaltung "Algorithmentheorie" sind Ihnen eine ganze Reihe weiterer NP-vollständiger Probleme bekannt, u.a.:

- IND SET (independent set)

- 0/1-IPROG (0/1-integer programming):

  Eingabe: Eine Liste linearer Ungleichungen mit rationalen Koeffizienten und Variablen $x_1, \ldots, x_n$

  Frage: Gibt es eine Belegung der Variablen $x_1, \ldots, x_n$ mit Werten aus $\{0,1\}$, so dass alle linearen Ungleichungen erfüllt sind?

- dHAMPATH (Existenz eines Hamiltonpfads bei gerichteten Graphen)

  Eingabe: Ein gerichteter Graph G

  Frage: Gibt es einen Weg in G, der jeden Knoten genau einmal besucht?

Es gibt auch einige natürliche Probleme (die in NP liegen), bei denen zur Zeit aber weder bekannt ist, dass sie in P liegen, noch dass sie NP-vollständig sind. Z.B.:

- <u>Graph Isomorphismus</u>

  <u>Eingabe</u>: Zwei Graphen G und H

  <u>Frage</u>: Ist $G \cong H$ (d.h. sind G und H isomorph?)

- <u>Faktorisierung</u>:

  <u>Eingabe</u>: Drei nat. Zahlen N, L, U

  <u>Frage</u>: Gibt es eine Primzahl p mit $U \leq p \leq L$, die N teilt?

Im Jahr 2002 wurde gezeigt*, dass das folgende Problem (überraschenderweise) in P liegt:

- <u>PRIMES</u> (die Eigenschaft, eine Primzahl zu sein)

  <u>Eingabe</u>: Eine natürliche Zahl N

  <u>Frage</u>: Ist N eine Primzahl?

(* von: Agrawal, Kayal, Saxena)

2.3 Entscheidungsprobleme vs. Suchprobleme

Idee:

- Entscheidungsprobleme: ja/nein - Antwort
- Suchprobleme:

  Ziel: Berechne eine gültige Lösung aus einem (i.d.R. großen) Raum möglicher Lösungskandidaten

  (z.B.: konstruiere eine erfüllende Belegung für eine gegebene aussagenlogische Formel)

Es gilt: Falls $P = NP$ ist, so können auch die "Suchproblem-Varianten" von NP-Problemen effizient (d.h. deterministisch in Polynomialzeit) gelöst werden.

Genauer:

Satz 2.13

Falls $P = NP$ ist, so gilt: Für jedes $L \in NP$ und jeden Verifizierer M für L (i.S. von Def. 2.1) gibt es eine (deterministische) Polynomialzeit-TM B s.d. f.a. $x \in \{0,1\}^*$ gilt:

- Falls $x \in L$, so gibt B bei Eingabe x ein Zertifikat y aus, s.d. $M(x,y) = 1$
- Falls $x \notin L$, so gibt B bei Eingabe x den Wert 0 aus

Beweis: ... Bekannt aus der Veranstaltung "Algorithmentheorie" (siehe auch [AB], Kapitel 2.5). Details: Übung! □

2.4 coNP, EXP und NEXP

Definition 2.14:

(a) Für $L \subseteq \{0,1\}^*$ ist $\overline{L} := \{0,1\}^* \setminus L$ das Komplement von L

Potenzmenge

(b) Ist K eine Menge von Sprachen, d.h. $K \subseteq \mathcal{P}(\{0,1\}^*)$

so ist

$$\mathrm{co}K := \{ \overline{L} : L \in K \}$$

die Komplementklasse von K.

Bemerkung 2.15

Insbesondere ist

$$\mathrm{coNP} = \{ \overline{L} : L \in \mathrm{NP} \}$$
$$= \{ L \subseteq \{0,1\}^* : \overline{L} \in \mathrm{NP} \}$$

Beachte: coNP ist nicht das Komplement von NP.
Insbesondere sieht man z.B. leicht, dass

$$\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{coNP}.$$

Fakt 2.16: Für jedes $L \subseteq \{0,1\}^*$ gilt: $L \in \mathrm{coNP} \iff$ es gibt ein Polynom $p: \mathbb{N} \to \mathbb{N}$ und eine Polynomialzeit-TM M, s.d. f.a. $x \in \{0,1\}^*$ gilt:

$$x \in L \iff \text{f.a. } u \in \{0,1\}^{p(|x|)} \text{ ist } M(\langle x, u\rangle) = 1.$$

Beweis: Übung. $\square$

Definition 2.17 (coNP-Vollständigkeit)

(a) $L' \subseteq \{0,1\}^*$ heißt **coNP-hart**, falls f.a. $L \in$ coNP gilt: $L \leq_p L'$.

(b) $L' \subseteq \{0,1\}^*$ heißt **coNP-vollständig**, falls $L' \in$ coNP und L' coNP-hart ist.

Beispiel 2.18

Die Sprache

$$\text{TAUTOLOGY} := \{ \varphi : \varphi \text{ ist eine allgemeingültige aussagenlogische Formel} \}$$

d.h. φ wird von jeder zu φ passenden Belegung erfüllt.

ist coNP-vollständig.

Beweis:

- TAUTOLOGY $\in$ coNP:
  Verwende dazu denselben Verifizierer M wie beim Nachweis, dass SAT $\in$ NP ist:
  Bei Eingabe von $\langle \varphi, z \rangle$ testet M, ob z eine Belegung ist, die φ erfüllt.

- coNP-Härte von TAUTOLOGY
  Sei $L \in$ coNP. Zu zeigen: $L \leq_p$ TAUTOLOGY
  Wegen $L \in$ coNP ist $\overline{L} \in$ NP. Daher $\overline{L} \leq_p$ SAT.
  Sei f eine Polynomialzeitreduktion von $\overline{L}$ auf SAT.
  D.h. f.a. $x \in \{0,1\}^*$ gilt:

$$x \in \overline{L} \iff \varphi_x := f(x) \in SAT$$

$x \notin L$ $\Leftrightarrow$

$\Updownarrow$

φ_x ist erfüllbar

$\Updownarrow$

$\neg\varphi_x$ ist nicht allgemeingültig

$\Updownarrow$

$\neg\varphi_x \notin$ TAUTOLOGY

Also: $x \in L \iff \neg\varphi_x \in$ TAUTOLOGY.

Daher ist die Abbildung g mit

$$g(x) := \neg\varphi_x \qquad (\forall \text{ a. } x \in \{0,1\}^*)$$

eine Polynomialzeit-Reduktion von L auf TAUTOLOGY.

$\square$

Bemerkung 2.13 (Vermutung: NP $\neq$ coNP)

Falls P = NP, so NP = coNP = P.

Daher gilt auch: Falls NP $\neq$ coNP, so P $\neq$ NP.

D.h.: NP $\neq$ coNP zu beweisen ist mindestens so schwer wie P $\neq$ NP zu beweisen

Umgekehrt ist durchaus denkbar, dass NP = coNP und trotzdem P $\neq$ NP.

Bisher ist nicht bekannt, ob NP $\neq$ coNP ist, aber – ähnlich wie bei P $\neq$ NP – glauben die meisten Komplexitätstheoretiker, dass wohl NP $\neq$ coNP ist.

EXP und NEXP

Zur Erinnerung: $\mathrm{EXP} = \bigcup_{c \geq 1} \mathrm{DTIME}\left(2^{(n^c)}\right)$

Analog: $\mathrm{NEXP} := \bigcup_{c \geq 1} \mathrm{NTIME}\left(2^{(n^c)}\right)$

NEXP ist die Klasse aller nichtdeterministisch in Polynomialzeit lösbaren Entscheidungsprobleme.

Wir wissen bereits:

$$P \subseteq NP \subseteq EXP \subseteq NEXP$$

D.h.: $EXP \neq NEXP$ zu beweisen ist mindestens so schwer wie $P \neq NP$ zu beweisen!

Satz 2.20

Falls $EXP \neq NEXP$, so $P \neq NP$.

(D.h.: Falls $P = NP$, so $EXP = NEXP$)

Beweis: Wir nehmen an, dass $P = NP$ gilt und zeigen, dass dann auch $EXP = NEXP$ ist.

Sei dazu $L \in NEXP$, d.h. es gibt ein $c \geq 1$ und eine NDTM M, die L in $2^{(n^c)}$ Schritten entscheidet. Zu zeigen: $L \in EXP$.

Wir betrachten dazu eine mit $2^{(|x|^c)}$ Einsen aufgefüllte (engl: "padded") Version von L:

$$L_{pad} := \left\{ \langle x, 1^{2^{(|x|^c)}} \rangle : x \in L \right\}$$

Beh 1: $L_{pad} \in NP$

Beweis: Eine NDTM N für L_{pad} kann bei Eingabe eines $y \in \{0,1\}^*$ wie folgt vorgehen:

1) Teste, ob y von der Form $\langle x, 1^{2^{(|x|^c)}} \rangle$ ist, für ein $x \in \{0,1\}^*$
   Falls nein: STOPP mit Ausgabe 0 (in q_{halt})
   Falls ja: weiter bei 2)
2) Lass M mit Eingabe x für $2^{(|x|^c)}$ Schritten laufen (mit den nichtdeterministischen Entscheidungen, die M trifft — nichtdeterministisch) und gib das aus, was M ausgibt.

Klar: N ist eine NDTM, deren Laufzeit polynomiell in der Länge der Eingabe y ist, und N entscheidet L_{pad}. Somit: $L_{pad} \in NP$

□ Beh 1

Gemäß unserer Annahme gilt $P = NP$.
Daher also: $L_{pad} \in P$, d.h. es gibt eine deterministische Polynomialzeit-TM $\tilde{N}$, die L_{pad} entscheidet. Eine det. Exponentialzeit-TM $\tilde{M}$, die L entscheidet, kann bei Eingabe eines $x \in \{0,1\}^*$ wie folgt vorgehen:

1) Konstruiere $\langle x, 1^{2^{(|x|^c)}} \rangle =: y$
   (das geht in Zeit exponentiell in $|x|$)
2) Lass $\tilde{N}$ mit Eingabe y laufen und gib die entsprechende Ausgabe aus
   (das geht in Zeit polynomiell in $|y|$, d.h. exponentiell in $|x|$).

Somit gilt: $L \in EXP$.

Definition 2.21

Eine Sprache $U \subseteq \{0,1\}^*$ heißt unär, falls $U \subseteq \{1\}^*$ ist.

Theorem 2.22 (Der Satz von Berman, 1978)

Falls es eine unäre NP-vollständige Sprache gibt,
so ist $P = NP$.

Beweis: siehe Übungsaufgaben.