Kapitel 3: Diagonalisierung

3.1 Zwei Zeithierarchie-Sätze

Die Laufzeit einer TM kann um beliebige konstante Faktoren beschleunigt werden:

Satz 3.1 (Lineare Beschleunigung)
Sei $T: \mathbb{N} \to \mathbb{N}$ mit $T(n) = \omega(n)$ und sei $\varepsilon > 0$.
Sei M eine TM, die eine Sprache $L \subseteq \{0,1\}^*$ in $T(n)$ Schritten entscheidet.
Dann gibt es auch eine TM, die L in nur $\varepsilon \cdot T(n)$ Schritten entscheidet.

Beweis: Übung. □

D.h.: Für unser TM-Berechnungsmodell gilt:
Wird die verfügbare Rechenzeit um einen konstanten Faktor vergrößert, so lassen sich nicht mehr Probleme mit dieser Rechenzeit lösen.

Der folgende Zeithierarchiesatz besagt: Wenn die Rechenzeit deutlich stärker als nur um einen konstanten Faktor vergrößert wird, so lassen sich tatsächlich mehr Probleme mit dieser Rechenzeit lösen.

Theorem 3.2 (Der Zeithierarchiesatz) (Hartmanis, Stearns, 1965)

Sei $T : \mathbb{N} \to \mathbb{N}$ eine zeitkonstruierbare Funktion und sei $t : \mathbb{N} \to \mathbb{N}$ eine Funktion mit

$$t(n) = o\left(\frac{T(n)}{\log(T(n))}\right)$$

Dann gilt:

$$\mathrm{DTIME}(t(n)) \subsetneq \mathrm{DTIME}(T(n))$$

Folgerung 3.3:

(a) F.a. $c \in \mathbb{N}$ gilt: $\mathrm{DTIME}(n^c) \subsetneq \mathrm{DTIME}(n^{c+1})$ und

$\mathrm{DTIME}(2^{(n^c)}) \subsetneq \mathrm{DTIME}(2^{(n^{c+1})})$

(b) $P \neq EXP$

(denn: $P \subseteq \mathrm{DTIME}(2^n) \subsetneq \mathrm{DTIME}(2^{(n^2)}) \subseteq EXP$)

Beweis von Theorem 3.2: Diagonalisierung!

Wir konstruieren eine TM D, die bei Eingabe $x \in \{0,1\}^*$ folgendes tut:

1) Simuliere die ersten $\frac{T(n)}{\log T(n)}$ Schritte der TM M_x mit Eingabe x.
2) Falls M_x innerhalb dieser Schrittzahl den Wert 0 (oder 1) ausgibt, so gib den umgekehrten Wert aus, d.h. 1 (oder 0). Ansonsten gib 0 aus.

Für Simulation nutzen wir die "effiziente universelle TM" aus Theorem 1.14 bzw. Bemerkung 1.15. Daher hält D für Eingaben der Länge n nach

$$\leq \frac{T(n)}{\log T(n)} \cdot \log\left(\frac{T(n)}{\log T(n)}\right) \leq \frac{T(n)}{\log T(n)} \cdot \log(T(n)) = T(n)$$

Schritten an.

Sei L_D die von D entschiedene Sprache.
Wir wissen: $L_D \in \mathrm{DTIME}(T(n))$

Behauptung: $L_D \notin \mathrm{DTIME}(t(n))$

Beweis: Angenommen doch, dann gibt es eine TM M, die L_D in $c \cdot t(n)$ Schritten entscheidet (wobei c eine geeignete Konstante ist). D.h. Bei Eingabe $x \in \{0,1\}^*$ hält M nach $\leq c \cdot t(|x|)$ Schritten an und gibt den Wert $D(x)$ aus.

Wegen $t(n) = o\left(\frac{T(n)}{\log T(n)}\right)$ gibt für alle hinreichend langen $x \in \{0,1\}^*$, dass

$$c \cdot t(|x|) < \frac{T(n)}{\log T(n)} \text{ ist.}$$

Wähle nun ein solchermaßen hinreichend langes $x \in \{0,1\}^*$ mit $M_x = M$. Dann gilt: $M = M_x$ bei Eingabe x hält nach $< T(n)/\log T(n)$ Schritten an und gibt den entgegengesetzten Wert aus, den D bei Eingabe x ausgibt. D.h: $M(x) = 1 \Leftrightarrow x \notin L_D$

↯ Widerspruch zur Annahme, dass M L_D entscheidet.

□

Ein ähnlicher Zeithierarchiesatz gilt auch für ndet TMs — der Beweis ist allerdings etwas aufwändiger.

Wir benutzen hier und im Folgenden die Notation:

Notation 3.4
Für $i \in \mathbb{N}_{>0}$ bezeichnet M_i die ndet TM M_α, wobei $\alpha \in \{0,1\}^*$ so, dass 1α die Binärdarstellung von i ist.
Somit ist $(M_i)_{i \in \mathbb{N}_{>0}} = (M_\alpha)_{\alpha \in \{0,1\}^*}$ eine Auflistung sämtlicher NDTMs.

Theorem 3.5 (Ndet Zeithierarchiesatz) (Cook, 1972)
Seien $t, T : \mathbb{N} \to \mathbb{N}$ zwei zeitkonstruierbare Funktionen mit $T(n) \geq t(n+1) \cdot (\log t(n+1))^2$ und $t(n) \geq n$ f.a. $n \in \mathbb{N}$.
Dann gilt: $\mathrm{NTIME}(t(n)) \subsetneq \mathrm{NTIME}(T(n))$

Folgerung 3.6
(a) F.a. $c \in \mathbb{N}$ gilt: $\mathrm{NTIME}(n^c) \subsetneq \mathrm{NTIME}(n^{c+1})$ und $\mathrm{NTIME}(2^{(n^c)}) \subsetneq \mathrm{NTIME}(2^{(n^{c+1})})$

(b) NP ≠ NEXP (denn: $NP \subseteq \mathrm{NTIME}(2^n) \subsetneq \mathrm{NTIME}(2^{(n^2)}) \subseteq NEXP$)

Skizze: $P \subseteq NP \subseteq EXP \subseteq NEXP$
mind. eine dieser beiden Inklusionen ist strikt.
mind. eine dieser beiden Inklusionen ist strikt.

Beweis von Theorem 3.5:

Diagonalisierung!

Beachte: Da es sich hier um ndet TMs handelt, kann man leider nicht einfach den Beweis von Theorem 3.2 anpassen.
(Warum? → Übung!)

Daher: "Faulenzer-Diagonalisierung"
(engl: lazy diagonalization):

Wähle eine stark wachsende Funktion $f: \mathbb{N} \to \mathbb{N}$, die trotzdem noch hinreichend effizient berechnet werden kann — etwa:

$$f(1) := 2 \quad \text{und}$$
$$f(i+1) := 2^{t(f(i)+1) \cdot \log(t(f(i)+1))}, \text{ f.a. } i \geq 1$$

Sei dann D eine NDTM, die bei Eingabe eines $x \in \{0,1\}^*$ folgendes tut:

1) Falls $x \notin \{1\}^*$: STOPP mit Ausgabe 0
Sonst: weiter mit 2)

2) Berechne $n := |x|$ (d.h.: $x = 1^n$) und die Werte $T(n)$, $t(n+1)$, $t(n+1) \cdot \log t(n+1)$

3) Berechne das $i \in \mathbb{N}$ mit $f(i) < n \leq f(i+1)$.

4) Falls $f(i) < n < f(i+1)$:

Simuliere $t(n+1) \cdot \log t(n+1)$ Schritte von M_i bei Eingabe 1^{n+1} (unter Verwendung nichtdeterministischer Entscheidungen) und gib die entsprechende Antwort aus und halte an

5) Falls $n = f(i+1)$: Sei $m := f(i)+1$ und $y := 1^m$

Akzeptiere $x = 1^n$ $\iff$ M_i lehnt $y = 1^m$ bei jeder Wahl nichtdeterministischer Entscheidungen nach $\leq t(m) \cdot \log t(m)$ Schritten ab.

Sei $L \subseteq \{0,1\}^*$ die von D nichtdeterministisch entschiedene Sprache

Behauptung 1: $L \notin NTIME(t(n))$

Behauptung 2: $L \in NTIME(T(n))$

$\Rightarrow$ Fertig mit Beweis von Theorem 3.5.

Beweis von Behauptung 1:

Angenommen, M ist eine NDTM, die L in $c \cdot t(n)$ Schritten entscheidet (wobei c eine Konstante ist)

Sei $i \in \mathbb{N}_{>0}$ hinreichend groß, so dass $M = M_i$ und $\log t(n+1) \geq c$ f.a. $n \geq f(i)$.

Daher simuliert D in Phase 4) und 5) die

gesamte Berechnung von M_i bei Eingabe von x der Länge $n > f(i)$ bzw bei Eingabe von y der Länge $m = f(i)+1$.

Somit gilt f.a. Eingaben $x = 1^n$ gemäß Phase 4):

Falls $f(i) < n < f(i+1)$, so

$$D(1^n) \overset{\text{Def } D}{=} M_i(1^{n+1})$$

M berechnet L $\Leftrightarrow$ $\Leftrightarrow$ $M = M_i$

$$M(1^n) \qquad M(1^{n+1}) \underset{M \text{ berechnet } L}{=} D(1^{n+1})$$

Also:

$$\text{(*)} \quad D(1^{f(i)+1}) = D(1^{f(i)+2}) = \cdots = D(1^{f(i+1)-1}) = D(1^{f(i+1)})$$

Andererseits gilt für $x = 1^n$ mit $n = f(i+1)$ gemäß Phase 5):

$$D(1^{f(i+1)}) \neq D(1^{f(i)+1})$$

↯ Widerspruch zu (*)

□ Beh 1

Beweis von Behauptung 2:

Schritte 1 und 2:
deterministisch möglich in Zeit $O(T(n))$
(da T, t zeitkonstruierbar)

Schritt 3:

Bei einer Eingabe der Länge n wird das gesuchte i wie folgt ermittelt:

a) $f(1) := 2$

b) for $i := 1, 2, 3, \ldots$ do:

b.1) Berechne $f(i+1) := 2^{t(f(i)+1) \cdot \log(t(f(i)+1))}$

b.2) Falls $f(i+1) > n$: STOPP.

Jeder Durchlauf durch Zeile b.1) lässt sich mit $O(t(n+1) \cdot \log(t(n+1)))$ Schritten realisieren.

Da in jedem Durchlauf eine 2-er-Potenz gebildet wird, sind wir nach $< \log t(n+1)$ Durchläufen fertig (tatsächlich sogar schon nach viel weniger Durchläufen).

Insgesamt werden in Schritt 3 also $O(T(n))$ Schritte gemacht

Schritt 4:

Unter Verwendung einer ndet. universellen TM in $O(t(n+1) \cdot (\log t(n+1))^2) = O(T(n))$ Schritten

Schritt 5:

Für ndet. Berechnungen der Länge $\leq t(n) \cdot \log t(n)$ gibt es insgesamt $2^{t(n) \log t(n)}$ viele verschiedene Folgen nichtdeterministischer Entscheidungen.

Beachte: Wegen $n = f(i)+1$ gilt gemäß unserer Wahl von f:

$$2^{t(n) \cdot \log t(n)} = f(i+1) = n \leq t(n)$$

Für jede dieser Folgen müssen Entscheidungen werden

$$t(m) \cdot \log t(m) < \log t(n)$$

Schritte von M_i simuliert. Durch Verwenden einer univ. TM geht das jeweils in $O(\log t(n) \cdot \log\log t(n))$ Schritten.

Insgesamt kann Phase 5) daher in

$$O\left(t(n) \cdot \log t(n) \cdot \log\log t(n)\right) = O(T(n))$$

Schritten durchgeführt werden.

D.h. D ist eine NDTM, die L in Zeit $O(T(n))$ entscheidet — also: $L \in NTIME(T(n))$

□ Beh 2

Insgesamt sind wir also fertig mit dem Beweis von Theorem 3.5.

□

3.2 Der Satz von Ladner

Theorem 3.6 (Der Satz von Ladner, 1975)

Falls $P \neq NP$, so gibt es eine Sprache L mit:

$L \in NP$, $L \notin P$ und L ist nicht NP-vollständig.

Beweis: Diagonalisierung!

Für jedes $H: \mathbb{N} \to \mathbb{N}$ sei SAT_H die folgendermaßen "aufgefüllte" ("gepaddete") Version von SAT:

$$SAT_H := \{ \psi 0 1^{(m^{H(m)})} : \psi \in SAT, m := |\psi| \}$$

Sei nun $H: \mathbb{N} \to \mathbb{N}$ wie folgt rekursiv definiert:

$H(0) := H(1) := 1$,

$H(n) :=$ die kleinste Zahl $i < \log\log n$, so dass f.a. $x \in \{0,1\}^*$ mit $|x| < \log n$ gilt:

M_i gibt bei Eingabe x nach $\leq i \cdot |x|^i$ Schritten den Wert $SAT_H(x)$ aus (d.h. 1 falls $x \in SAT_H$; 0 sonst)

— bzw. den Wert $\log\log n$, falls kein solches $i < \log\log n$ existiert.

Beachte: H ist wohldefiniert, da bei der Wahl von $H(n)$ nur auf SAT_H-Instanzen der Länge $< n$ betrachtet werden.

Behauptung 0:
Es gibt einen det. Polynomialzeit-Algorithmus, der bei Eingabe eines Worts der Länge n in Zeit $poly(n)$ die Zahl $H(n)$ berechnet.

Beweis: Übung!

Unter Verwendung von Beh 0 sieht man leicht, dass $SAT_H \in NP$ ist.

Behauptung 1:

(a) $SAT_H \in P \Rightarrow$ es gibt ein $C \in \mathbb{N}$ s.d. $H(n) \leq C$ f.a. $n \in \mathbb{N}$.

(b) $SAT_H \notin P \Rightarrow H(n) \xrightarrow{n \to \infty} \infty$.

Beweis:

(a): Sei $SAT_H \in P$ und sei M eine (det.) TM, die SAT_H in $\leq d \cdot n^d$ Schritten löst, für eine Konstante d.

Sei $i > d$ s.d. $M = M_i$.

Gemäß der Definition von H gilt f.a. $n > 2^{2^i}$, dass $H(n) < i$.

Daher fertig mit $C := \max\{i, H(0), H(1), \dots, H(2^{2^i})\}$

(b): Angenommen, $H(n) \not\xrightarrow{n \to \infty} \infty$.

Dann gibt es ein $C \in \mathbb{N}$ und eine unendliche Folge von Zahlen $0 \leq n_0 < n_1 < n_2 < \cdots$ s.d. f.a. $j \in \mathbb{N}$ gilt: $H(n_j) \leq C$.

Dann gibt es auch ein $i \leq C$ s.d. $H(n_j) = i$ für unendlich viele $j \in \mathbb{N}$ gilt.

Gemäß unserer Wahl von H gilt dann für jedes solche j:

F.a. $x \in \{0,1\}^*$ mit $|x| < \log n_j$ gibt M_i bei Eingabe x nach $\leq i \cdot |x|^i$ Schritten den Wert $SAT_H(x)$ aus.

Daher entscheidet M_i die Sprache SAT_H in $i \cdot n^i$ Schritten — d.h.: $SAT_H \in P$

□ Beh 1.

Behauptung 2:
Falls $P \neq NP$, so ist $SAT_H \notin P$

Beweis:
Angenommen, $SAT_H \in P$. Gemäß Beh 1(a) gibt es dann ein $C \in \mathbb{N}$ s.d. $H(n) \leq C$ f.a. $n \in \mathbb{N}$

Somit ist

$$SAT_H = \{ \varphi 0 1^{(n^{H(n)})} : \varphi \in SAT,\ n = |\varphi|,\ H(n) \leq C \}$$

Dann gibt es eine Polynomialzeit-Reduktion von SAT auf SAT_H, die bei Eingabe φ einfach $n = |\varphi|$ ermittelt, $H(n)$ berechnet (vgl. Beh. 0), $1^{(n^{H(n)})}$ berechnet (das geht in Zeit poly(n), da $H(n) \leq C$) und dann $\varphi 0 1^{(n^{H(n)})}$ ausgibt.
Aus $SAT_H \in P$ folgt dann: $SAT \in P$.
Da SAT NP-vollständig ist, gilt dann $P = NP$ □ Beh 2

Behauptung 3
Falls $P \neq NP$, so ist SAT_H nicht NP-vollständig.

Beweis: Es sei $P \neq NP$.
- Angenommen, SAT_H ist NP-vollständig. Dann gibt es eine Polynomialzeit-Reduktion f von SAT auf SAT_H. Sei $i \in \mathbb{N}$ s.d. f in $\leq n^i$ Schritten berechnet wesden kann.

- Außerdem wissen wir aus Beh 2 bereits, dass $SAT_H \notin P$ ist — und mit Beh 1(b) folgt daher, dass $H(n) \xrightarrow{n\to\infty} \infty$. Insbes. ist $H(n) > i$ für alle hinreichend großen n.

- Die Polyn.zeit-Reduktion f bildet eine SAT-Instanz φ der Länge n auf eine SAT_H-Instanz $\psi 01^{(m^{H(m)})}$ ab, wobei $m := |\psi|$ ist.

  Insbes. gilt:

  $m^{H(m)} < n^i$, da $f(\varphi)$ in n^i Schritten berechnet werden kann.

  Für alle hinreichend großen n muss dann

  $m < n^{1/2}$ sein

  (denn ansonsten würde für hinreichend große n mit $\frac{1}{2}H(m) > i$ gelten: $m^{H(m)} \geq \left(n^{1/2}\right)^{H(n^{1/2})} = n^{\frac{1}{2}\cdot H(m)} > n^i$ ↯).

  Somit gilt f.a. hinreichend großen n:
  f bildet SAT-Instanzen φ der Länge n ab auf SAT_H-Instanzen $\psi 01^{(m^{H(m)})}$, wobei $|\psi| = m < n^{1/2}$, s.d. gilt:

  $\varphi \in SAT \iff \psi \in SAT$.

- Durch wiederholtes Anwenden von f erhält man eine Folge $\psi_1, \psi_2, \psi_3, \ldots$ von SAT-Instanzen,

für die gilt:

(1) $\varphi \in SAT \iff \psi_1 \in SAT \iff \psi_2 \in SAT \iff \psi_3 \in SAT \iff \dots$

und

(2) $|\psi_1| < n^{\frac{1}{2}}$, $|\psi_2| < (n^{\frac{1}{2}})^{\frac{1}{2}}$, $|\psi_3| < n^{\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}}$, und allgemein f.a. j: $|\psi_j| < n^{(1/2^j)}$

Frage: Für welches j gilt: $n^{(1/2^j)} \leq \log n$?

Antwort:

$$n^{1/2^j} \leq \log n \iff 2^{(\log n)\cdot\frac{1}{2^j}} \leq 2^{\log\log n}$$

$$\iff (\log n)\cdot\frac{1}{2^j} \leq \log\log n$$

$$\iff \frac{\log n}{\log\log n} \leq 2^j \impliedby \log\log n \leq j$$

Somit gilt: $\log\log n$-faches iteriertes Anwenden der Polynomialzeit-Reduktion f liefert eine SAT-Instanz $\psi' := \psi_{\log\log n}$ der Länge $\leq \log n$,

s.d. $\varphi \in SAT \iff \psi' \in SAT$.

In ψ' können höchstens $|\psi'| \leq \log n$ verschiedene Variablen vorkommen. Ein det. Algorithmus, der nacheinander alle $2^{\log n} = n$ Variablenbelegungen durchprobiert, kann in poly(n) Schritten ermitteln ob $\psi' \in SAT$ ist. Zum Erzeugen von ψ' werden auch nur poly(n) Schritte benötigt, da

die in Zeit n^i berechenbare Reduktion insgesamt nur $\log\log n$ mal aufgerufen wird, um $\psi^i = \psi_{\log\log n}$ zu erzeugen.

Damit erhalten wir einen det. Polynomialzeit-Algorithmus, der SAT löst. Also $SAT \in P$ und daher $P = NP$.

↳ Widerspruch zur Annahme, dass $P \neq NP$ ist

□ Beh 3

Insgesamt ist für $L := SAT_{\#}$ damit Theorem 3.6. bewiesen

□

3.3 Orakel-TMen und die Grenzen der Diagonalisierung

Definition 3.7

a) Eine Orakel-TM ist eine (det.) TM, die ein spezielles sog. Orakel-Band besitzt sowie drei spezielle Zustände q_{query}, q_{yes}, q_{no}.
Bei Eingabe $x \in \{0,1\}^*$ mit Orakel $O \subseteq \{0,1\}^*$ geht M wie eine herkömmliche TM vor. Aber jedesmal, wenn M im Zustand q_{query} ist, ist sie im nächsten Schritt im Zustand q_{yes} bzw q_{no}, je nach dem ob das Wort, das gerade auf dem Orakel-Band

steht, ihr Sprache O gehört oder nicht.

(Beachte: Unabhängig davon, wie O aussieht, zählt jedes solche "Befragen des Orakels" als nur ein Schritt von M).

Wir schreiben $M^O(x)$, um die Ausgabe von M bei Eingabe x mit Orakel O zu bezeichnen.

(b) Nichtdeterministische Orakel-TMen werden analog definiert.

Definition 3.8 (P^O, NP^O)

Für $O \subseteq \{0,1\}^*$ ist

- P^O die Klasse aller Sprachen $L \subseteq \{0,1\}^*$, die durch eine det. Orakel-TM mit Orakel O in polynomiell vielen Schritten entschieden werden können
- NP^O die Klasse aller Sprachen $L \subseteq \{0,1\}^*$, die durch eine ndet. Orakel-TM mit Orakel O in polynomiell vielen Schritten entschieden werden können

Beispiel 3.9

(a) $\overline{SAT} \in P^{SAT}$ durch eine Orakel-TM M, die bei Eingabe einer Formel φ diese auf's Orakelband schreibt, dann in den Zustand q_{query} geht und 0 bzw 1 ausgibt, je nach dem ob M dann in den Zustand q_{yes} bzw q_{no} geht.

(b) Für $O \in P$ ist $P^O = P$, denn:

"$\supseteq$": klar.

"$\subseteq$": Sei M eine det. polyn.zeit Orakel-TM, die eine Sprache $L \in P^O$ entscheidet, und sei N eine det. polyn.zeit TM, die O entscheidet. Jedesmal wenn M das Orakel O befragt, starten wir die TM N. Die dadurch erhaltene TM M' ist dann eine det. polyn.zeit TM, die L entscheidet. ∎

(c) Sei

$$EXPCOM := \{ \langle M, x, 1^t \rangle : M \text{ ist eine det TM, } x \in \{0,1\}^*, t \in \mathbb{N}, \text{ s.d. } M \text{ bei Eingabe } x \text{ nach} \leq 2^t \text{ Schritten den Wert 1 ausgibt} \}$$

Dann gilt:

$$P^{EXPCOM} = NP^{EXPCOM} = EXP \quad \left(\overset{\text{Def}}{=} \bigcup_{c \geq 1} DTIME(2^{(n^c)}) \right)$$

Beweis:

$EXP \subseteq P^{EXPCOM}$:

Sei $L \in EXP$, d.h. es gibt ein $c \geq 1$ und eine det TM M, die L in $2^{(n^c)}$ Schritten entscheidet.

Somit gilt f.a. $x \in \{0,1\}^*$:

$x \in L \Leftrightarrow M$ akzeptiert x nach $\leq 2^{(|x|^c)}$ Schritten

$\Leftrightarrow \langle M, x, 1^t \rangle \in EXPCOM$, für $t = |x|^c$.

Somit ist $L \in P^{EXPCOM}$ durch eine det. polyn.zeit TM M', die bei Eingabe x zunächst den Wert $t = |x|^c$

berechnet, dann das Wort $\langle M, x, 1^{\ell} \rangle$ aufs Orakel-Band schreibt und dann 1 bzw 0 ausgibt je nach dem ob M' nach Befragen des Orakels im Zustand q_{yes} oder q_{no} ist.

$P^{EXPCOM} \subseteq NP^{EXPCOM}$:

Klar.

$NP^{EXPCOM} \subseteq EXP$:

Sei M eine ndet. polynzeit TM, die mit Orakel EXPCOM eine Sprache $L \subseteq \{0,1\}^*$ entscheidet. Sei $c \in \mathbb{N}$ s.d. M bei Eingaben der Länge n stets nach $\leq n^c$ Schritten anhält. Dann gibt es $\leq 2^{(n^c)}$ verschiedene Folgen nichtdeterministischer Entscheidungen, nach denen M verfahren kann. Außerdem kann die Sprache EXPCOM durch eine det. Exponentialzeit-TM entschieden werden. Mittels Durchprobieren aller Folgen ndet. Entscheidungen und Simulation von M mit Berechnung jeder Orakel-Anfrage in Exponentialzeit erhält man eine det. Exponentialzeit-TM, die L entscheidet. Also: $L \in EXP$.

□

Theorem 3.10 (Satz von Baker, Gill und Solovay, 1975)

Es gibt Orakel $A, B \subseteq \{0,1\}^*$ so dass

$$P^A = NP^A \quad \text{und} \quad P^B \neq NP^B$$

Beweis:

Mit $A := EXPCOM$ folgt aus Bsp 2.9 (c), dass $P^A = NP^A$.

Zur Konstruktion von B s.d. $P^B \neq NP^B$ betrachten wir für jedes $B \subseteq \{0,1\}^*$ die unäre Sprache

$$U_B := \{ 1^n : n \in \mathbb{N}, \text{ es } x \in \{0,1\}^n \text{ mit } x \in B \}.$$

Für jedes $B \subseteq \{0,1\}^*$ ist $U_B \in NP^B$, denn bei Eingabe eines Worts der Form 1^n kann eine ndet. Orakel-TM zunächst ein Wort $x \in \{0,1\}^n$ raten und dann das Orakel befragen, ob $x \in B$ liegt.

Wir konstruieren nun ein spezielles $B \subseteq \{0,1\}^*$, s.d. $U_B \notin P^B$ (dann folgt $P^B \neq NP^B$).

Dazu sei für jedes $i \in \mathbb{N}_{>0}$ M_i die det. Orakel-TM, die durch die Binärdarstellung von i (ohne die führende 1) repräsentiert wird.

Wir konstruieren B in Stufen $i = 1, 2, 3, \ldots$, so dass die i-te Stufe bewirkt, dass es eine Zahl $n_i \geq i$ gibt, so dass M_i^B auf Eingaben der Länge n_i die Sprache U_B nicht in $\leq 2^{n_i}/10$ Schritten entscheidet.

Insgesamt folgt daraus dann, dass es keine det. Orakel-TM gibt, die U_B in Polynomialzeit entscheidet.

Konstruktion von B in Stufen $i = 1, 2, 3, \dots$:

Zu Beginn der Konstruktion ist $B = \emptyset$. In jeder Stufe $i \in \mathbb{N}_{>0}$ wird für eine endliche Anzahl von Worten $x \in \{0,1\}^*$ festgelegt, ob $x \in B$ oder $x \notin B$ ist.

Stufe i ($i \in \mathbb{N}_{>0}$):

Sei $n_i \geq i$ groß genug, so dass in den Stufen $j < i$ für kein Wort $x \in \{0,1\}^{n_i}$ festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist.

Lass M_i mit Eingabe 1^{n_i} für $2^{n_i}/10$ Schritte laufen. Wann immer M_i dabei das Orakel für ein Wort $x \in \{0,1\}^*$ befragt, tue folgendes:

- falls in den Stufen $j < i$ bereits festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist, antworte entsprechend mit ja bzw. nein.
- falls in keiner der Stufen $j < i$ festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist, lege fest, dass $x \notin B$ ist

Fall 1: Falls diese Berechnung die Eingabe 1^{n_i} nach $\leq 2^{n_i}/10$ Schritten akzeptiert, so lege fest, dass kein Wort der Länge n_i zu B gehört. Insbes. gilt dann:

$1^{n_i} \notin U_B$.

Falls diese Berechnung die Eingabe 1^{n_i} nicht nach $\leq 2^{n_i}/10$ Schritten akzeptiert, so wähle ein beliebiges Wort $x \in \{0,1\}^{n_i}$ aus, für das bisher noch nicht festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist (ein solches Wort gibt es, da $|\{0,1\}^{n_i}| = 2^{n_i}$ ist, die Berechnung aber höchstens $2^{n_i}/10 < 2^{n_i}$ Orakel-Anfragen macht). Lege fest, dass $x \in B$ ist. Insbes gilt dann:

$1^{n_i} \in U_B$.

In beiden Fällen gilt:

$1^{n_i} \in U_B \iff M_i^B$ akzeptiert die Eingabe 1^{n_i} nicht nach $\leq 2^{n_i}/10$ Schritten. (*)

Angenommen, U_B wird durch eine det. Orakel-TM M in $\leq n^c$ Schritten entschieden (d.h. $U_B \in P^B$).

Wähle dann i hinreichend groß so dass $M = M_i$ und $n_i^c \leq 2^{n_i}/10$.

Aus (*) folgt dann, dass U_B nicht durch M_i^B entschieden wird. ↯ Widerspruch. □

Bemerkung 3.11:

(a) Die Beweise der beiden Zeithierarchiesätze Theorem 3.2 und Theorem 3.5 lassen sich leicht auf Orakel-TMen übertragen (Details: Übung!), so dass man für jedes Orakel $O \subseteq \{0,1\}^*$ erhält:

- $\mathrm{DTIME}^O(n^c) \subsetneq \mathrm{DTIME}^O(n^{c+1})$ (f.a. $c \geq 1$)
- $\mathrm{DTIME}^O(2^{(n^c)}) \subsetneq \mathrm{DTIME}^O(2^{(n^{c+1})})$
- $\mathrm{NTIME}^O(n^c) \subsetneq \mathrm{NTIME}^O(n^{c+1})$ (f.a. $c \geq 2$)
- $\mathrm{NTIME}^O(2^{(n^c)}) \subsetneq \mathrm{NTIME}^O(2^{(n^{c+1})})$
- $P^O \neq EXP^O$, $NP^O \neq NEXP^O$.

(b) Da es Orakel A, B mit $P^A = NP^A$ und $P^B \neq NP^B$ gibt (laut Theorem 3.10), kann

$P \neq NP$ (oder $P = NP$)

daher nicht allein mit Methoden (z.B. Diagonalisierung) bewiesen werden, die sich auch direkt auf Orakel-TMen übertragen lassen.