



---

# Komplexitätstheorie

---

## Kapitel 4: Mehr Ressourcen, mehr Möglichkeiten?

# Einleitung

Bei P vs NP vergleicht man **unterschiedliche Maschinenmodelle**

Wenn man ein Maschinenmodell und eine Ressource fixiert gilt intuitiv:

Mehr Ressourcen, mehr Probleme lösbar!!

In diesem Kapitel:

- Für natürliche Fälle ist das meist auch richtig (Hierarchie-Theoreme)
- Es gibt bemerkenswerte Ausnahmen (Gap-Theoreme)

# Kapitel 4

## Hierarchie-Theoreme

# Hierarchietheoreme

Hierarchietheoreme:

- Klasse von Resultaten, die Komplexitätsklassen **separieren**, mit denen also die Echtheit von Inklusionen nachgewiesen werden kann
- Betrachten Klassen, die auf **demselben Maschinenmodell** und **derselben Ressource** beruhen
- Basieren auf **Diagonalisierungsbeweisen**, ähnlich zu Beweisen von Unentscheidbarkeit (z.B. Halteproblem)

Zum warm werden: ein kurzer Exkurs zur Unentscheidbarkeit

# Unentscheidbarkeit

## Theorem.

Es gibt unentscheidbare Sprachen.

Diagonalisierungsbeweis:

- Modifiziere Kodierung von DTMs als Wörter, so dass **jedes** Wort DTM repräsentiert (zum Beispiel durch Annahme einer Default-DTM)  
Für jedes  $w \in \{0, 1\}^*$  sei  $\mu(w)$  die kodierte DTM
- Betrachte Tabelle für Akzeptanz von Wörtern durch DTMs
- Deren **Diagonale** besteht aus Positionen  $M, w$  mit  $\mu(w) = M$
- Sprache  $L_D$ : Komplementiere die Werte auf der Diagonalen
- Führe Widerspruchsbeweis für Unentscheidbarkeit von  $L_D$  ●

# Hierarchietheoreme

Wir beweisen nur ein sehr spezielles Hierarchietheorem

Zur Erinnerung:  $\text{ExpTime} := \bigcup_{i \geq 1} \text{DTime}(2^{\mathcal{O}(n^i)})$ .

Theorem.

$P \subsetneq \text{EXPTIME}$

Diagonalisierung soll Problem in  $\text{EXPTIME} \setminus P$  liefern:

- in ExpTime: betrachte nur Akzeptanz in exponentiell vielen Schritten
- Intuitiv sollte das Problem immernoch nicht in P sein

Wir nehmen an, dass jede DTM durch **unendlich viele** Wörter kodiert wird,  
(Stelle z.B. sicher, dass  $\mu(0^*w) = \mu(w)$  für alle  $w \in \Sigma^*$ )

# P und ExpTime

Sprache  $L_D$  implementiert Diagonalisierung und Komplementierung:

**Lemma.**

$L_D := \{w \in \Sigma^* \mid \mu(w) \text{ akzeptiert } w \text{ nicht in } \leq 2^{|w|} \text{ Schritten} \}$   
ist in  $\text{EXPTIME} \setminus P$

$L_D \notin P$ : Widerspruchsbeweis

- Annahme: es gibt  $p$ -zeitbeschränkte DTM  $M$  mit  $L(M) = L_D$
- Wähle  $w_i$  mit  $\mu(w_i) = M$  und  $2^{|w_i|} \geq p(|w_i|)$
- $w_i \in \mu(w_i)$  und  $w_i \notin \mu(w_i)$  führt beides zu Widerspruch ●

# P und ExpTime

Sprache  $L_D$  implementiert Diagonalisierung und Komplementierung:

## Lemma.

$L_D := \{w \in \Sigma^* \mid \mu(w) \text{ akzeptiert } w \text{ nicht in } \leq 2^{|w|} \text{ Schritten} \}$   
ist in  $\text{EXPTIME} \setminus \text{P}$

$L_D \in \text{EXPTIME}$ : DTM  $M$

- simuliert  $\mu(w)$  auf Eingabe  $w$
- zählt die Schritte, die schon simuliert wurden
- verwirft, wenn  $\mu(w)$  in  $\leq 2^{|w|}$  Schritten akzeptiert
- akzeptiert, wenn  $\mu(w)$  in  $\leq 2^{|w|}$  Schritten verwirft
- akzeptiert, wenn  $\mu(w)$  mehr als  $2^{|w|}$  Schritte macht





## P und ExpTime

$L_D$  ist unnatürliches Problem. Natürliche  $L \in \text{ExpTime} \setminus P$  via Vollständigkeit:

### Definition ExpTime-Härte, ExpTime-Vollständigkeit

Problem  $L$  ist

- *ExpTime-hart* wenn  $L' \leq_p L$  für alle  $L' \in \text{ExpTime}$ ;
- *ExpTime-vollständig* wenn  $L$  EXPTIME-hart und in EXPTIME.

Offensichtlich: wenn EXPTIME-hartes  $L \in P$ , dann  $L_D \in P$ ; also:

**Lemma.**

Wenn  $L$  EXPTIME-hart, dann  $L \notin P$ .

# P und ExpTime

ExpTime-vollständige Probleme:

- das Wortproblem für polyplatzbeschränkte **alternierende** TMs typisch für ExpTime, ähnlich wie SAT und 3SAT für NP
- manche Spielprobleme  
z.B. Existenz von Gewinnstrategien in Dame Spielen  
(bei beliebig grossem Brett)
- manche Probleme in der Logik  
z.B.: Erfüllbarkeit von Formeln der Spezifikationslogik CTL
- manche Datenbankprobleme  
z.B. Inklusion zwischen XPath-Ausdrücken, für manche Fragmente von XPath 1.0 und 2.0

# Zeithierarchiesatz

Das gerade gezeigte Resultat ist Spezialfall eines wesentlich generelleren Satzes

## Definition Zeitkonstruierbar

Funktion  $t : \mathbb{N} \rightarrow \mathbb{N}$  heißt *zeitkonstruierbar* wenn es eine DTM  $M$  gibt mit  $\text{time}_M(w) = t(|w|)$  für alle  $w \in \Sigma^*$ .

Intuitiv: Funktionen, die mit den Ressourcenvorgaben berechnet werden können, die sie selbst machen (“vernünftige” Funktionen)

Natürliche Funktionen sind i.d.R. zeitkonstruierbar, z.B.:

- $n^i$  für alle  $i \geq 1$
- $2^n$

# Zeithierarchiesatz

## Theorem (Zeithierarchie).

Für jede zeitkonstruierbare Funktion  $t_2$  und jede Funktion  $t_1$  mit  $t_2 \in \omega(t_1 \cdot \log(t_1))$  gilt:  $\text{DTime}(t_1) \subsetneq \text{DTime}(t_2)$ .

$f \in \omega(g)$  bedeutet “ $f$  wächst schneller als  $g$ ”

Konsequenzen dieses Resultats z.B.:

- $\text{DTime}(n^i) \subsetneq \text{DTime}(n^{i+1})$  für alle  $i \geq 1$   
(aber keine natürlichen Probleme in  $\text{DTime}(n^{i+1}) \setminus \text{DTime}(n^i)$  bekannt)
- $\text{DTime}(n^i) \subsetneq \text{P}$  für alle  $i \geq 0$
- $2\text{-ExpTime} := \bigcup_{i \geq 1} \text{DTime}(2^{2^{O(n^i)}})$ ,  $3\text{-ExpTime} := \bigcup_{i \geq 1} \text{DTime}(2^{2^{2^{O(n^i)}}})$ , etc.

Dann  $k\text{-ExpTime} \subsetneq (k+1)\text{-ExpTime}$  für alle  $k \geq 1$

# Zeithierarchiesatz

## Theorem (Zeithierarchie).

Für jede zeitkonstruierbare Funktion  $t_2$  und jede Funktion  $t_1$  mit  $t_2 \in \omega(t_1 \cdot \log(t_1))$  gilt:  $\text{DTime}(t_1) \subsetneq \text{DTime}(t_2)$ .

Unterschiede im Beweis:

- Konstruiere TM  $M$ , die einige Schritte von  $\mu(w)$  simuliert, komplementär akzeptiert und dabei selbst exakt  $t_2(n)$  Schritte macht
- Dazu wird Zeitkonstruierbarkeit verwendet: gleichzeitiges Simulieren einer TM, die exakt  $t_2(|w|)$  Schritte macht
- Verwende  $L(M)$  als  $L_D$ , dann trivial:  $L(M) \in \text{DTime}(t_2)$
- $L(M) \notin \text{DTime}(t_1)$ : gleiche Idee, vorsichtigere Analyse Zeitverbrauch

# Hierarchiesätze

Weitere Hierarchiesätze existieren, z.B. nicht-deterministische Zeit:

**Theorem (Nicht-deterministische Zeithierarchie).**

Für jede zeitkonstruierbare Funktion  $t_2$  und jede Funktion  $t_1$  mit  $t_2(n) \in \omega(t_1(n + 1))$  gilt:  $\text{NTime}(t_1) \subsetneq \text{NTime}(t_2)$ .

Beweis etwas anders aber Konsequenzen wie in det. Fall, z.B.

- $\text{NTime}(n^i) \subsetneq \text{NTime}(n^{i+1})$  für alle  $i \geq 1$ ;
- $\text{NP} \subsetneq \text{NExpTime} := \bigcup_{i \geq 1} \text{NTime}(2^{\mathcal{O}(n^i)})$

# Kapitel 4

## Gap-Theoreme

# Gap Theorem

Scheinbar paradox: es gibt trotzdem beliebig große “Lücken” zwischen Komplexitätsklassen

## Theorem (Gap-Theorem).

Für jede Funktion  $g : \mathbb{N} \rightarrow \mathbb{N}$  gibt es eine berechenbare Funktion  $t : \mathbb{N} \rightarrow \mathbb{N}$  so dass  $\text{DTime}(t) = \text{DTime}(g(t))$ .

Zum Beispiel:

- $g = n^2$ .

Es gibt Funktion  $t$  mit  $\text{DTime}(t) = \text{DTime}(t^2)$ .

- $g = 2^n$ .

Es gibt Funktion  $t$  mit  $\text{DTime}(t) = \text{DTime}(2^t)$ .



## Beweis: Vorbereitung

Sei

- $M_0, M_1, M_2, \dots$  eine Aufzählung aller Turingmaschinen
- $t_i(n)$  die Zeitkomplexität von  $M_i$  auf Eingaben der Länge  $n$ , d.h.:

wenn  $M_i$  auf jeder Eingabe  $w$  der Länge  $n$  stoppt:

$$t_i(n) = \max\{\text{time}_{M_i}(w) \mid w \text{ Eingabe der Länge } n\}$$

wenn  $M_i$  auf mindestens einer Eingabe  $w$  der Länge  $n$  nicht stoppt:

$$t_i(n) = \infty$$

Für gegebene Funktion  $g$ :

Wir wollen  $t$  finden, so dass **keine** Funktion  $t_i$  zwischen  $t$  und  $g(t)$  liegt ●

## Beweis (fast)

Für jedes  $n \geq 0$  wählen wir

$$t(n) = \max\{t_i(n) \mid i \leq n \text{ und } t_i(n) < \infty\} + 1$$

### Lemma

$$\text{DTime}(t) = \text{DTime}(g(t))$$

Problem:

- es gibt keinen Grund, warum dieses  $t$  **berechenbar** sein sollte
- insbesondere können wir für gegebenes  $i$  und  $n$  nicht entscheiden, ob  $t_i(n) = \infty$  oder nicht

## Beweis (jetzt aber)

Definiere modifizierte Funktion  $t$ , berechne  $t(n)$  wie folgt:

Starte mit  $t(n) = n + 1$

while  $t(n) \leq t_i(n) \leq g(t(n))$  for some  $i \leq n$  do

$t(n)++$

Intuition:  $t(n)$  liegt nicht mehr unbedingt über allen  $t_i(n)$  mit  $i \leq n$

der Bereich  $[t(n), g(t(n))]$  kann auch "Lücke" zwischen allen diesen  $t_i$  sein

### Lemma

1. Die Berechnung von  $t(n)$  terminiert.
2.  $DTime(t) = DTime(g(t))$



## Nachbemerkung

Der scheinbare Widerspruch zum Hierarchietheorem ist leicht lösbar:

- die Funktion  $t$  ist zwar berechenbar, aber sie wächst extrem schnell
- insbesondere ist sie **nicht zeitkonstruierbar**

Das Gap-Theorem basiert also auf der Existenz “extremer” Funktionen

Wir interessieren uns meist für “normale” Funktionen, da spielt das Gap-Theorem keine wichtige Rolle.

Analoge Theoreme gibt es für nichtdeterministische Zeit