Komplexitätstheorie

SoSe 2018
Thomas Schneider

Kapitel 4: Mehr Ressourcen, mehr Möglichkeiten?

Homepage der Vorlesung: http://tinyurl.com/ss18-kt

Einleitung

Bei „P versus NP“ vergleicht man unterschiedliche Maschinenmodelle.

Wenn man ein Maschinenmodell und eine Ressource fixiert, gilt intuitiv:

Mehr Ressourcen $\Rightarrow$ mehr Probleme lösbar

In diesem Kapitel:

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

Kapitel 4

NEXT

4.1 Hierarchietheoreme

4.2 Gap-Theoreme

Hierarchietheoreme

Hierarchietheoreme:

- sind Resultate, die Komplexitätsklassen trennen,
  mit denen also die Echtheit von Inklusionen nachgewiesen werden kann.
- betrachten Komplexitätsklassen,
  die auf demselben Maschinenmodell und derselben Ressource beruhen.
- basieren auf Diagonalisierungsbeweisen,
  ähnlich zu Beweisen von Unentscheidbarkeit (z.B. Halteproblem).

Zum Aufwärmen: eine kurze Wiederholung zur Unentscheidbarkeit

Unentscheidbarkeit

Theorem 4.1

Es gibt unentscheidbare Sprachen.

Beweis per Diagonalisierung:

- 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

T4.1

Hierarchietheoreme

Wir beweisen nur ein sehr spezielles Hierarchietheorem.

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

Theorem 4.2

P $\subsetneq$ ExpTime

Diagonalisierung soll Problem in ExpTime $\setminus$ P liefern:

- „in ExpTime":
  betrachte nur Akzeptanz in max. exponentiell vielen Schritten
- Intuitiv sollte das Problem immer noch nicht in P sein

P und ExpTime

Wir nehmen an, dass jede DTM durch unendlich viele Wörter kodiert wird.

(Stelle z. B. sicher, dass $\mu(0^* w) = \mu(w)$ für alle $w \in \Sigma^*$)

Sprache L_D implementiert Diagonalisierung und Komplementierung:

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

T4.2

Lemma 4.3

$L_D \in$ ExpTime \ P

$L_D \notin$ P: Widerspruchsbeweis

Angenommen es gebe p-zeitbeschränkte DTM M mit $L(M) = L_D$.

Wähle w_i mit $\mu(w_i) = M$ und $2^{|w_i|} \geq p(|w_i|)$.

$w_i \in L\big(\mu(w_i)\big)$ und $w_i \notin L\big(\mu(w_i)\big)$ führt beides zu Widerspruch

T4.3

P und ExpTime

Wir nehmen an, dass jede DTM durch unendlich viele Wörter kodiert wird.

(Stelle z. B. sicher, dass $\mu(0^*w) = \mu(w)$ für alle $w \in \Sigma^*$)

Sprache L_D implementiert Diagonalisierung und Komplementierung:

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

Lemma 4.3

$L_D \in \text{ExpTime} \setminus \text{P}$

$L_D \in$ 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

T4.4

ExpTime-Härte und -Vollständigkeit

L_D ist unnatürliches Problem.

Natürliche $L \in$ ExpTime $\setminus$ P via Vollständigkeit:

Definition 4.4 (ExpTime-Härte, ExpTime-Vollständigkeit)

Problem L ist

- *ExpTime-hart*, wenn $L' \leq_p L$ für alle $L' \in$ ExpTime;
- *ExpTime-vollständig*, wenn L ExpTime-hart und in ExpTime.

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

Lemma 4.5

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

ExpTime-Härte und -Vollständigkeit

ExpTime-vollständige Probleme:

- das Wortproblem für polyplatzbeschränkte alternierende TMs
  – ist prototypisch 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
  oder Konzept-Erfüllbarkeit bzgl. TBoxen in der Beschreibungslogik $\mathcal{ALC}$
- 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 des wesentlich genereller en Zeithierarchiesatzes.

Vorbetrachtung:

Definition 4.6 (zeitkonstruierbare Fkt.)

Eine Funktion $t : \mathbb{N} \to \mathbb{N}$ heißt *zeitkonstruierbar,*
wenn es eine DTM M gibt, so dass für alle $w \in \Sigma^*$ gilt:

$$\text{time}_M(w) = t(|w|)$$

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 4.7 (Zeithierarchie)

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

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

Konsequenzen dieses Resultats z. B.:

- $\mathsf{DTime}(n^i) \subsetneq \mathsf{DTime}(n^{i+1})$ für alle $i \geq 1$

  (aber keine natürlichen Probleme in $\mathsf{DTime}(n^{i+1}) \setminus \mathsf{DTime}(n^i)$ bekannt)

- $\mathsf{DTime}(n^i) \subsetneq \mathsf{P}$ für alle $i \geq 0$

- $k\mathsf{ExpTime} \subsetneq (k+1)\mathsf{ExpTime}$ für alle $k \geq 1$, wobei

  $2\mathsf{ExpTime} := \bigcup_{i \geq 1} \mathsf{DTime}(2^{2^{\mathcal{O}(n^i)}})$, $3\mathsf{ExpTime} := \bigcup_{i \geq 1} \mathsf{DTime}(2^{2^{2^{\mathcal{O}(n^i)}}})$ etc.

T4.5

Zeithierarchiesatz

Theorem 4.7 (Zeithierarchie)

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

Beweisskizze: (vgl. Bew. Lemma 4.3)

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

T4.6

Hierarchiesätze

Weitere Hierarchiesätze existieren, z. B. nichtdeterministische Zeit:

Theorem 4.8 (Nichtdeterministischer Zeithierarchiesatz).
Für jede zeitkonstruierbare Funktion t_2 und jede Funktion t_1 mit $t_2(n) \in \omega(t_1(n+1))$ gilt: $\mathsf{NTime}(t_1) \subsetneq \mathsf{NTime}(t_2)$

Beweis etwas anders,
hat aber analoge Konsequenzen wie im deterministischen Fall, z. B.

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

Gap-Theorem

Scheinbar paradox:

es gibt trotzdem beliebig große „Lücken“ zwischen Komplexitätsklassen

Theorem 4.9 (Gap-Theorem)

Für jede berechenbare Funktion $g : \mathbb{N} \to \mathbb{N}$
gibt es eine berechenbare Funktion $t : \mathbb{N} \to \mathbb{N}$,
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

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

  Wenn M_i auf jeder Eingabe w der Länge n stoppt, dann

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

  Wenn M_i auf mind. 1 Eingabe w der Länge n nicht stoppt, dann

  $$t_i(n) = \infty$$

Für gegebene Funktion g :

Wollen t finden, so dass keine Funktion t_i zwischen t und $g(t)$ liegt.

T4.7

Beweis (fast)

Für jedes $n \geq 0$ wählen wir

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

T4.8

Lemma 4.10

DTime(t) = DTime$(g(t))$

T4.9

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.

Für gegebenes n berechne $t(n)$ wie folgt:

$t(n) \leftarrow n + 1$

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

  $t(n)$++

Intuition: $t(n)$ liegt nicht mehr unbedingt über allen $t_i(n)$ mit $i \le n$;
Intervall $[t(n), g(t(n))]$ kann auch „Lücke“ zwischen allen diesen t_i sein.

Lemma 4.11

1. t ist berechenbar.
2. $\mathrm{DTime}(t) = \mathrm{DTime}(g(t))$

T4.10

Nachbemerkung

Widerspruch zwischen Hierarchie- und Gap-Theorem?

Aus HT folgt z. B.: DTime$(n) \subsetneq$ DTime(n^2)

Aus GT folgt z. B.: es gibt Fkt. t mit DTime$(t) =$ DTime(t^2)

… ist leicht auflösbar:

- Die Funktion t ist zwar berechenbar, aber sie wächst extrem schnell.
- Insbesondere ist sie nicht zeitkonstruierbar!

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

Analoge Theoreme gibt es für nichtdeterministische Zeit.

Übersicht Vorlesung

Kapitel 1: Einführung

Kapitel 2: Turingmaschinen

Kapitel 3: P vs. NP

Kapitel 4: Mehr Ressourcen, mehr Möglichkeiten?

NEXT

Kapitel 5: Platzkomplexität

Kapitel 6: Schaltkreise

Kapitel 7: Orakel