Beschleunigung, Kompression und Hierarchien

Dr. Eva Richter

22. Juni 2012

Einige typische Komplexitätsklassen

Raumklassen	Zeitklassen
$L = DSPACE(log)$	$REALTIME = DTIME(id)$
$NL = NSPACE(log)$	$LINTIME = DTIME(Lin)$
$LINSPACE = DSPACE(Lin)$	$P = DTIME(Pol)$
$NLINSPACE = NSPACE(Lin)$	$NP = NTIME(Pol)$
$PSACE = DSPACE(Pol)$	$E = DTIME(2^{Lin})$
$NPSPACE = NPSPACE(Pol)$	$NE = NTIME(2^{Lin})$
$EXPSPACE = DSPACE(2^{Pol})$	$EXP = DTIME(2^{Pol})$
$NEXPSPACE = NSPACE(2^{Pol})$	$NEXP = NTIME(2^{Pol})$

Es geht immer schlimmer

Satz

Für jede total berechenbare Funktion $t : \mathbb{N} \to \mathbb{N}$ gibt es ein Problem A_t, so dass $A_t \notin DTIME(t)$.

Beweis:

- Sei $M_0, M_1 \dots$ eine Aufzählung aller DTM.
- $$A_t := \{0^i \mid M_i \text{ akzeptiert } 0^i \text{ nicht in } t(i) \text{ Schritten}\}$$
- Annahme: $A_t \in DTIME(t)$, d.h. es gibt j mit $L(M_j) = A_t$ und $time_{M_j}(n) \leq t(n)$.

$$0^j \in A_t \Leftrightarrow M_j \text{ akzeptiert } 0^j \text{ nicht in } t(j) \text{ Schritten}$$
$$0^j \notin L(M_j) = A_t$$

- Widerspruch, also gilt $A_t \notin DTIME(t)$.

Sparsamer Umgang mit Ressourcen

- Wie stark muss eine Ressource vergrößert werden, damit echt mehr berechnet werden kann ?
- Für Komplexitätsklasse $DTIME(t_1)$ fragen wir, wie viel stärker eine Funktion t_2 wachsen muss, damit die Ungleichheit $DTIME(t_1) \neq DTIME(t_2)$ gilt.
- Sätze für Raum und Zeit besagen, dass ein linearer Zuwachs der gegebenen Ressourcenfunktion *nicht* genügt.

Beschleunigungssatz

Satz

Für jede total berechenbare Funktion $t : \mathbb{N} \to \mathbb{N}$ mit $id \in o(t)$ gilt:

$$DTIME(t) = DTIME(Lin(t)).$$

Beweis:

- Sei $time_M(n) \leq t(n)$ für eine DTM M mit $L(M) = A$. Konstruieren eine DTM N mit $\varphi_M = \varphi_N$, die m-mal schneller als M ist für eine Konstante $m > 1$.
- N tut m Schritte von M in einem Schritt.
- Bandalphabet von N kodiert mehrere Zeichen von M in einem Zeichen, N hat mehr Zustände als M.
- Wir konstruieren N in zwei Phasen: in der ersten wird die Eingabe in die komprimierte Kodierung überführt und in der zweiten die Arbeit von M simuliert.

Als Maschinenmodell nutzen wir eine Zweibandturingmaschine, die auf einem Band nur liest auf dem anderen arbeitet.

Beweis Phase 1

- N kodiert das Eingabewort $w_1, \dots w_n$ mit $w_i \in \Sigma$, indem sie es in Blöcke der Länge m aufteilt, wobei der i-te Block durch das Wort $\beta_i = w_{1+(i-1)m} w_{2+(i-1)m} \dots w_{im}$ repräsentiert wird, d.h.

$$w = \beta_1 \beta_2 \dots \beta_{k+1} \text{ mit } k = \lfloor n/m \rfloor$$

- N schreibt folgende Kodierung auf ihr Band: $(\square^m, \beta_1, \beta_2)(\beta_1, \beta_2, \beta_3) \dots (\beta_{k-2}, \beta_{k-1}, \beta_k)(\beta_{k-1}, \beta_k, \square^m)$
- Jedes Tripel wird als ein Symbol von N aufgefasst. Die erste Phase erfordert $n + k = (1 + \frac{1}{m})n$ Schritte.

Raumkompressionssatz

Satz

Für jede total berechenbare Funktion $s : \mathbb{N} \to \mathbb{N}$ gilt:

$$DSPACE(s) = DSPACE(Lin(s)).$$

Beweis Raumkompressionssatz

Beweis:

- zeigen, dass $DSPACE(2 \cdot s) \subseteq DSPACE(s)$.
- Sei M eine DTM, mit $space_M(n) \leq 2s(n)$, konstruieren N mit $L(M) = L(N)$ und $space_N(n) \leq s(n)$.
- M hat Alphabet Γ und soll nur auf geraden Bandzellen nach links dürfen.
- N hat mehr Zustände und Bandalphabet $\Gamma \times \Gamma$.
- Unterteile Band von M in Zweierblöcke, numeriere Zellen von N mit $(2i - 1, 2i)$ und trage Inhalte a und b von M als ein Zeichen ab ein.
- N simuliert das Verhalten von M auf Eingabe w, indem es dasselbe tut wie M, nur dass sie den Kopf erst bewegt, wenn M die Blockgrenze überschreitet. Auf diese Weise berechnet N dieselbe Funktion wie M, aber braucht nur Raum $s(n)$.

$\square$

Beweis Phase 2

- für $a = a_1 \dots a_l$ auf Band von M, dann bekommt N

$$(\square^m, \alpha_1, \alpha_2)(\alpha_1, \alpha_2, \alpha_3) \dots (\alpha_{z-2}, \alpha_{z-1}, \alpha_z)(\alpha_{z-1}, \alpha_z, \square^m)$$

- N simuliert die m Schritte von M: falls M ein Zeichen aus α_j liest, so steht der Kopf von N auf $(\alpha_{j-1}, \alpha_j, \alpha_{j+1})$ Nach m Schritten steht der Kopf von M auf Zelle von α_{j-1}, α_j oder α_{j+1}), d.h. N kann alles in einem Schritt ausführen und geht auf

$$\begin{aligned} &(\alpha_{j-2}, \alpha_{j-1}, \alpha_j), \text{ falls } M \text{ eine Bandzelle im Block } \alpha_{j-1} \text{ liest;} \\ &(\alpha_{j-1}, \alpha_j, \alpha_{j+1}), \text{ falls } M \text{ eine Bandzelle im Block } \alpha_j \text{ liest} \\ &(\alpha_j, \alpha_{j+1}, \alpha_{j+2}), \text{ falls } M \text{ eine Bandzelle im Block } \alpha_{j+1} \text{ liest} \end{aligned}$$

- Akzeptiert oder verwirft M die Eingabe w, so tut dies auch N. Also ist $L(M) = L(N)$.
- Phase 2 erfordert höchstens $\lceil t(n)/m \rceil$ Schritte

Beweis–Abschätzung der Laufzeit von N

- Voraussetzung $id \in o(t)$ heißt $\forall c > 0$ gilt $n <_{ae} c \cdot t(n)$
- Laufzeit Phase 1: $(1 + \frac{1}{m})n$, Phase 2: $\lceil t(n)/m \rceil$, zusammen:

$$\left(1 + \frac{1}{m}\right) n + \left\lceil \frac{t(n)}{m} \right\rceil <_{ae} \left(1 + \frac{1}{m}\right) \frac{1}{m(1+\frac{1}{m})} t(n) + \left\lceil \frac{t(n)}{m} \right\rceil$$
$$\leq \left\lceil \frac{2t(n)}{m} + 1 \right\rceil$$

- erste Ungleichung folgt mit der spezifischen Konstante

$$c = \frac{1}{m(1+\frac{1}{m})} = \frac{1}{m+1}$$

- Damit haben wir gezeigt, dass eine beliebige lineare Beschleunigung möglich ist.

Beispiel

- Sei $t(n) = d \cdot n$ für eine Konstante $d > 1$ die Laufzeitbeschränkung von N habe die Laufzeit

$$T(n) = \left(1 + \frac{1}{m}\right) n + \frac{t(n)}{m} = \left(1 + \frac{1}{m}\right) n + \frac{d \cdot n}{m} = \left(1 + \frac{d+1}{m}\right) n$$

- Wenn m sowohl n als auch $t(n)$ teilt, folgt aus $d > 1$ bei $m > (d+1)/(d-1)$

$$T(n) < d \cdot n = t(n)$$

und somit eine echte Beschleunigung.
- $t(n) = d \cdot n$ mit $d > 1$ wächst nicht echt schneller als die Identitätsfunktion, d.h. Beispiel zeigt, dass die Voraussetzung $id \in o(t)$ etwas stärker ist, als notwendig.
- Mit $d = 1$ also $t = id$ würde der Beweis jedoch nicht funktionieren.

Satz (Rosenberg)

REALTIME $\neq$ *LINTIME*

Hierarchiesätze

- Wie stark muss eine Ressource vergrößert werden, damit echt mehr berechnet werden kann?
- Haben bewiesen, dass linearer Zuwachs der Ressourcenfunktion nicht genügt, um echt größere Klassen zu erhalten.
- Wenn $s_2 \in O(s_1)$, dann wächst s_2 nicht stark genug, um s_1 an Wirkung zu übertreffen.
- Statt $\exists c > 0$ mit $s_2(n) \leq_{ae} c \cdot s_1(n)$ betrachten wir s_1, s_2 mit

$$\forall c > 0 \text{ gilt } s_2(n) >_{io} c \cdot s_1(n) \Leftrightarrow s_2 \succ_{io} s_1$$

Raum-und Zeitkonstruierbarkeit

Definition

Seien f, s und t total berechenbare Funktionen.

- *s heißt raumkonstruierbar, falls es eine DTM M gibt, so dass für alle n gilt: M benötigt bei einer beliebigen Eingabe der Länge n nicht mehr als $s(n)$ Bandzellen, um das Wort $\sharp 1^{s(n)-2}\$$ auf das Band zu schreiben, wobei $\sharp$ und $\$$ spezielle Symbole zur Markierung des linken und rechten Randes sind. Man sagt, M hat den Raum $s(n)$ ausgelegt.*
- *f heißt konstruierbar in der Zeit t, falls es eine DTM M gibt, so dass für jedes n gilt: M arbeitet bei einer beliebigen Eingabe der Länge n genau $t(n)$ Takte und schreibt dabei das Wort $\sharp 1^{f(n)-2}\$$ auf das Band. Man sagt, t ist zeitkonstruierbar, falls t in der Zeit t konstruierbar ist.*

Raumhierarchiesatz

Satz

Falls gilt, dass $s_1 \prec_{io} s_2$ und falls s_2 raumkonstruierbar ist, dann gilt:

$$DSPACE(s_2) \not\subseteq DSPACE(s_1)$$

- Beweis nur für den Fall $s_1 \geq log$
- Wir konstruieren eine Menge A in der Differenz $DSPACE(s_1) \setminus DSPACE(s_2)$ durch Diagonalisierung.
- Brauchen eine Aufzählung $M_0, M_1, M_2, \ldots$ aller Einbandturingmaschinen.
- Wir definieren eine DTM N mit einem Eingabeband und drei Arbeitsbändern.

Raumhierarchiesatz–Beweis

$N =$

Auf Eingabe $w \in \{0,1\}^*$ der Länge n

1. N legt den Raum $s_2(n)$ auf allen Arbeitsbändern aus.
2. Sei $w = 1^i y$, wobei $0 \leq i \leq n$ und $y \in \{\epsilon\} \cup 0\{0,1\}^*$
3. N interpretiert i als Maschinennummer und schreibt das geeignet kodierte Programm von M_i auf das erste Arbeitsband. Falls das Programm größer als $s_2(n)$ ist, reject. Sonst simuliert N das Verhalten von M_i auf w auf dem zweiten Band.
4. Das dritte Band enthält einen Binärzähler, der anfangs den Wert 0 enthält und in jedem Schritt der Simulation von M_i um eins erhöht wird. Ist die Simulation von M_i beendet bevor der Zähler überläuft, accept wenn M_i akzeptiert, sonst reject.

Technische Erläuterungen

- Der Zähler garantiert, dass N immer anhält.
- Es gibt eine Konstante c_i, so dass die Simulation von M_i auf dem zweiten Band im Raum höchstens in $c \cdot space_{M_i}$ gelingt. Der Grund dafür ist, dass N in der Lage sein muss, jede DTM M_i zu simulieren. Wenn M_i für ein i insgesamt z_i Zustände und l_i Symbole in ihrem Bandalphabet hat, dann kann N diese Symbole und Zustände binär als Wörter der Länge $\lceil \log z_i \rceil$ bzw. $\lceil \log l_i \rceil$ kodieren. Diese Kodierung verursacht zusätzlich konstante Raumkosten für die simulierende Maschine N, wobei die Konstante c_i nur von M_i abhängt.

Beweis Raumhierarchiesatz Fortsetzung I

- $A =_{def} L(M) \in DSPACE(s_2)$, **Annahme** $A \in DSPACE(s_1)$
- Dann würde eine DTM M_i mit $A = L(M_i)$ existieren und $space_{M_i}(n) \leq s_1(n)$.
- Wegen $s_2(n) \succ_{io} c \cdot s_1(n)$, d.h. $\forall c > 0$ gilt $s_2(n) >_{io} c \cdot s_1(n)$ gibt es c und unendlich viele $n_1, n_2 \ldots \in \mathbb{N}$ mit $s_2(n_k) > c \cdot s_1(n_k)$

Wähle n_j aus, so dass

(i) Das Programm von M_i kann im Raum $s_2(n_j)$ berechnet und auf das zweite Arbeitsband geschrieben werden.

(ii) Die Simulation von M_i auf $1^i 0^{n_j - i}$ gelingt im Raum $s_2(n_j)$.

(iii) $time_{M_i}(n_j) \leq 2^{s_2(n_j)}$.

Beweis Raumhierarchiesatz Fortsetzung II

Wähle n_j aus, so dass

(i) Das Programm von M_i kann im Raum $s_2(n_j)$ berechnet und auf das zweite Arbeitsband geschrieben werden.
Größe des Programms von M_i hängt nicht von der Eingabe der Maschine ab

(ii) Die Simulation von M_i auf $1^i 0^{n_j - i}$ gelingt im Raum $s_2(n_j)$.
wegen $c_i \cdot space_{M_i}(n_j) \leq c_i \cdot s_1(n_j) < s_2(n_j)$, wobei c_i die Konstante ist, die durch die Kodierung von M_i entsteht.

(iii) $time_{M_i}(n_j) \leq 2^{s_2(n_j)}$.
nach Raum-Zeit-Satz, folgt für $s_1 \geq \log$

$$time_{M_i}(n_j) \leq 2^{d \cdot space_{M_i}(n_j)} \leq 2^{d \cdot s_1(n_j)} < 2^{s_2(n_j)}$$

Für $w = 1^i 0^{n_j - i}$ gilt: $w \in A \Leftrightarrow N$ akzeptiert $w \Leftrightarrow M_i$lehnt ab.
D.h. $A \neq L(M_i)$; Widerspruch zur Annahme, also
$A \notin DSPACE(s_1)$.

Die Ernte

Beweis liefert sogar ein noch stärkeres Resultat:

Folgerung

$DSPACE(s_2) \subsetneq \bigcup_{s_1 \prec_{io} s_2} DSPACE(s_1)$

und

Folgerung

Falls $s_1 \leq s_2$ und $s_1 \prec_{io} s_2$ und s_2 ist raumkonstruierbar, dann gilt:

$$DSPACE(s_1) \subset DSPACE(s_2)$$

Sei $POLYLOGSPACE =_{def} \bigcup_{k \geq 1} DSPACE((\log n)^k)$, dann gilt:

L $\subset$ **POLYLOGSPACE** $\subset$ **LINSPACE** $\subset$ **PSPACE** $\subset$ **EXPSPACE**.

Zeithierarchiesatz

Satz

Sei $t_2 \geq id$ und $t_1 <_{io} t_2$ und sei t_2 in der Zeit $t_2 \log t_2$ konstruierbar, so gilt

$$DTIME(t_2 \log t_2) \not\subseteq DTIME(t_1).$$

Beweisidee:

- beruht auf Diagonalisierung
- starte mit Aufzählung der DTM
- die diagonalisierende DTM N, die in $t_2 \log t_2$ arbeitet, muss jede Maschine M_i „schlagen“
  d.h. wenn M_i in t_1 arbeitet, dann haben M und N verschiedene Sprachen

Zeithierarchiesatz–Folgerungen

Folgerung

1. Falls $t_1 \leq t_2 \log t_2$ und $t_2 \geq id$ und $t_1 \prec_{io} t_2$ und t_2 in der Zeit $t_2 \log t_2$ konstruierbar ist, dann gilt $DTIME(t_1) \subset DTIME(t_2 \log t_2)$.
2. Für jede Konstante $k > 0$ gilt $DTIME(n^k) \subset DTIME(n^{k+1})$ und $DTIME(2^{k \cdot n}) \subset DTIME(2^{(k+1) \cdot n})$.
3. $P \subset E \subset EXP$