

Komplexitätstheorie

SoSe 2019

Jean Christoph Jung, Thomas Schneider

Kapitel 4: Platzkomplexität

Homepage der Vorlesung: <http://tinyurl.com/ss19-kt>

Kapitel 4

NEXT



4.1 Platzkomplexitätsklassen

- 4.2 Polynomieller Platzverbrauch (PSPACE)
- 4.3 Zeitkomplexität versus Platzkomplexität
- 4.4 Der Satz von Savitch
- 4.5 PSPACE-Härte und -Vollständigkeit
- 4.6 Logarithmischer Platz (LogSpace)

Einleitung

Ab jetzt betrachten wir **Platzkomplexität**.

Genauer:

- der temporäre Zwischenspeicher, der während der Berechnung verwendet wird (Datenstrukturen, Rekursionsstack etc.)
- **im Fall von TMs:** die verwendeten Bandzellen (außer denen für die Eingabe!)

Platzverbrauch kann in der Praxis so kritisch sein wie Zeitverbrauch:

- Geringer Zeitverbrauch hilft nicht, wenn Speicher vorher erschöpft.
- Dies passiert nicht selten bei Algorithmen, die schwierige Probleme lösen.

Wesentlicher Unterschied zur Zeitkomplexität:

Platz kann man wiederverwenden, Zeit nicht.

Platzkomplexität

Um Eingabe und „Zwischenspeicher“ zu trennen, nehmen wir für alle DTMs / NTMs ein **dediziertes Eingabeband** an:

- Eingabeband wird **nur gelesen**, nicht beschrieben. (d. h. TM muss per Konvention das auf diesem Band gelesene Symbol wieder zurückschreiben)
- TM kann sich auf dem Eingabeband nach links und rechts bewegen, also die Eingabe auch **mehrfach lesen**.
- Im Zusammenhang mit Platzkomplexität ist eine **k-Band-TM** also eine TM mit **k Arbeitsbändern + 1 Eingabeband**.

Dies ermöglicht es uns, auch Platzbeschränkungen s mit $s(n) \leq n$ zu betrachten!

Aufgrund Ihrer Bedeutung für NP betrachten wir auch **NTMs**.

Platzkomplexität

Definition 4.1 (platzbeschränkte TM)

Für eine k -Band-DTM und $w \in \Sigma^*$ schreiben wir $\text{space}_M(w) = n$, wenn die (eindeutige) Berechnung von M auf w n **Bandzellen** verwendet. alle Bänder aufsummiert

Für eine k -Band-NTM und $w \in \Sigma^*$ ist $\text{space}_M(w)$ analog definiert, aber bezüglich der Berechnung von M auf w , die den meisten Platz verbraucht.

Sei $s : \mathbb{N} \rightarrow \mathbb{N}$ monoton wachsende Funktion.

M ist s -**platzbeschränkt**, wenn $\text{space}_M(w) \leq s(n)$ für alle w der Länge n .

Nun ist:

$$\text{DSpace}_k(s) := \{L \subseteq \Sigma^* \mid \exists \mathcal{O}(s)\text{-platzbeschränkte terminierende } k\text{-Band-DTM } M \text{ mit } L(M) = L\}$$

$$\text{NSpace}_k(s) := \{L \subseteq \Sigma^* \mid \exists \mathcal{O}(s)\text{-platzbeschränkte terminierende } k\text{-Band-NTM } M \text{ mit } L(M) = L\}$$

T4.1

Mehrband-TMs

Auch bei Platzkomplexität brauchen wir uns um die **Anzahl Bänder** **keine Gedanken** zu machen:

Satz 4.2 (Bandreduktion, Platzkomplexität)

Für alle $k \geq 1$ und s gilt: $\text{DSpace}_k(s) \subseteq \text{DSpace}_1(s)$

Beweis: exakt derselbe wie für Zeitkomplexität (verwende ein Band mit mehreren Spuren)

Beachte: im Gegensatz zu Zeitkomplexität **kein quadratischer Blowup!**

Definition 4.3 (DSpace, NSpace)

$$\text{DSpace}(s) := \bigcup_{k \geq 1} \text{DSpace}_k(s); \quad \text{NSpace}(s) := \bigcup_{k \geq 1} \text{NSpace}_k(s)$$

Kapitel 4

4.1 Platzkomplexitätsklassen

NEXT

4.2 Polynomieller Platzverbrauch (PSpace)

4.3 Zeitkomplexität versus Platzkomplexität

4.4 Der Satz von Savitch

4.5 PSpace-Härte und -Vollständigkeit

4.6 Logarithmischer Platz (LogSpace)

PSpace

Analog zu polynomieller Zeit kann man **polynomiellen Platz** betrachten.

Mehr als polynomiell viel Platz anzunehmen ist **nicht realistisch**.

Aber auch polynomieller Platz ist schon „recht viel“:

- Betrachte Eingabe der Größe 10.000
- kubischer Zeitverbrauch (n^3): 5 Minuten auf 3-GHz-Prozessor
- kubischer Platzverbrauch: 1 Terabyte Speicher nötig!

Entsprechende Komplexitätsklasse:

$$\text{PSpace} := \bigcup_{i \geq 1} \text{DSpace}(n^i)$$

Ein „typisches“ Problem in PSpace:

Gültigkeit von quantifizierten Booleschen Formeln (QBFs)

So zu verstehen:

- „Boolesche Formeln“ = AL-Formeln
- „quantifiziert.“ zusätzliche **Quantoren** für Wahrheitswerte, als **Präfix**

Definition 4.4 (QBF)

Eine *quantifizierte Boolesche Formel (QBF)* hat die Form

$$Q_1 p_1 \cdots Q_n p_n \varphi$$

wobei $Q_i \in \{\forall, \exists\}$ und φ eine AL-Formel mit Variablen aus der Menge $\{p_1, \dots, p_n\}$.

AL-Formeln dürfen hier auch die **Konstanten 0,1** enthalten.

T4.2

Definition 4.5 (Gültigkeit einer QBF)

QBF $Q_1 p_1 \cdots Q_n p_n \varphi$ ist *gültig*, wenn

- $n = 0$ und φ (welches dann variabelnfrei ist) zu 1 auswertet.
- $Q_1 = \exists$ und $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/0]$ **oder** $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/1]$ gültig
- $Q_1 = \forall$ und $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/0]$ **und** $Q_2 p_2 \cdots Q_n p_n \varphi[p_1/1]$ gültig

Dabei steht $\varphi[p/0]$ für „ φ mit p durch 0 ersetzt“,
 $\varphi[p/1]$ für „ φ mit p durch 1 ersetzt“

$$\text{z. B. } (p_1 \vee p_2) \rightarrow (p_2 \vee p_3)[p_2/0] = (p_1 \vee 0) \rightarrow (0 \vee p_3)$$

Überprüfen von Gültigkeit: durch **Auswertungsbaum**

T4.3

Auswertungs bäume

Auswertungsbaum für QBF $\psi = Q_1 p_1 \cdots Q_n p_n \varphi$:

- binärer Baum der Tiefe n
- Wurzel korrespondiert zu ψ
- Übergang zu Nachfolger entspricht Elimination eines Quantors durch Festlegen des Variablenwertes
- Blätter entsprechen also AL-Formeln ohne Variablen

Erfolgreicher Auswertungsbaum:

- „Hochpropagieren“ der Auswertung:
- \exists -Quantoren $\hat{=}$ **or**; \forall -Quantoren $\hat{=}$ **and** (**and/or-Baum**)
- **Wurzel** muss zu 1 evaluieren

Erfolgreicher Auswertungsbaum ist „**Beweis**“ für Gültigkeit wie in der Definition von NP, aber **exponentiell groß!**

QBF

Theorem 4.6

QBF ist in PSpace.

Idee:

- Verwende rekursive Prozedur, die sich aus Def. von Gültigkeit ergibt.
- Rekursionstiefe ist **linear**, für jeden Rekursionsschritt wird linear viel Speicher gebraucht.
- Ergibt **quadratischen Platzbedarf**: man kann Speicher **wiederverwenden**, wenn man in anderen Ast des (exp. großen) Rekursionsbaums absteigt.

Theorem 4.6

QBF ist in PSpace.

Algorithmus:

```

Function valid( $Q_1 p_1 \cdots Q_n p_n \varphi$ ):
  if  $n = 0$  then //  $\varphi$  ist AL-Formel, variablenfrei
    | evaluiere  $\varphi$  und gib Ergebnis zurück
   $x \leftarrow \text{valid}(Q_2 p_2 \cdots Q_n p_n \varphi[p_1/0])$ 
   $x' \leftarrow \text{valid}(Q_2 p_2 \cdots Q_n p_n \varphi[p_1/1])$ 
  if  $Q_1 = \exists$  then return  $x \vee x'$  else return  $x \wedge x'$ 

```

- Rekursionstiefe beschränkt durch Länge des Quantorenpräfixes
- Pro Aufruf speichern: Argument (QBF) des momentanen Aufrufs plus Rücksprungadresse (ob x oder x' berechnet wurde)

Insgesamt also quadratischer Platz.

- 4.1 Platzkomplexitätsklassen
- 4.2 Polynomieller Platzverbrauch (PSpace)
- NEXT** → **4.3 Zeitkomplexität versus Platzkomplexität**
- 4.4 Der Satz von Savitch
- 4.5 PSpace-Härte und -Vollständigkeit
- 4.6 Logarithmischer Platz (LogSpace)

Zeit versus Platz

Natürliche Frage:

- Wie verhält sich PSpace zu P, NP und ExpTime?
- (allgemeiner) Wie verhält sich Platzkomplexität zu Zeitkomplexität?

Wir geben im Folgenden **2 Antworten**.

Zeit versus Platz (1)

„Was man in **Zeit** f berechnen kann, kann man auch in **Platz** f berechnen, sogar für **nichtdeterministische Zeit** und **deterministischen Platz**.“

Theorem 4.7

Für alle (monoton wachsenden) $f : \mathbb{N} \rightarrow \mathbb{N}$ gilt:

$$\text{NTime}(f) \subseteq \text{DSpace}(f)$$

Beweis.

Sei $L \in \text{NTime}(f)$ und M f -zeitbeschränkte NTM mit $L(M) = L$.

Für Konfiguration $uqav$ von M und Übergang $\alpha = (q, a, q', a', B) \in \Delta$ sei **Schritt**($uqav, \alpha$) die Nachfolgekonfig. von $uqav$ bei Anwendung von α .

Folgender Algorithmus gibt **true** zurück

gdw. es eine Berechnung von M gibt, die mit $uqav$ startet und **akzeptiert**.

Zeit versus Platz (1)

```
Function  $acc_M(uqav)$ :  
  if  $q = q_{acc}$  then return true  
  if  $q = q_{rej}$  then return false  
  forall  $\alpha = (q, a, q', a', B) \in \Delta$  do  
    if  $acc_M(\text{Schritt}(uqav, \alpha))$  then return true  
  return false
```

Leicht zu sehen:

$acc_M(q_0w)$ antwortet true gdw. M akzeptiert w

Zeit versus Platz (1)

```
Function  $acc_M(uqav)$ :  
  if  $q = q_{acc}$  then return true  
  if  $q = q_{rej}$  then return false  
  forall  $\alpha = (q, a, q', a', B) \in \Delta$  do  
    if  $acc_M(\text{Schritt}(uqav, \alpha))$  then return true  
  return false
```

Platzverbrauch:

- Rekursionstiefe beschränkt durch $f(|w|)$
 - Platzbedarf pro Rekursionsschritt ist $\mathcal{O}(f(|w|))$:
 - $|uqav| \in \mathcal{O}(f(|w|))$
(pro Rechenschritt nur $\mathcal{O}(1)$ viele Zellen beschrieben)
 - (q, a, q', a', B) ist konstant groß (mit M gegeben)
- ↪ Platzverbrauch zunächst $\mathcal{O}(f(n)^2)$

Zeit versus Platz (1)

```
Function  $acc_M(uqav)$ :  
  if  $q = q_{acc}$  then return true  
  if  $q = q_{rej}$  then return false  
  forall  $\alpha = (q, a, q', a', B) \in \Delta$  do  
    if  $acc_M(\text{Schritt}(uqav, \alpha))$  then return true  
  return false
```

Linearer Platzverbrauch ist wie folgt erreichbar:

- $uqav$ nur $1 \times$ in globaler Variable speichern
- Übergänge α auf diese globale Konfiguration anwenden
- durchgeführte Übergänge weiterhin auf Stack speichern (je konstant groß)
- bei Rückkehr aus rekursivem Abstieg: letzten Übergang „rückgängig machen“ (d. h. rückwärts auf globale Konfig. anwenden)

Zeit versus Platz (1)

Theorem 4.7 (wiederholt)

Für alle (monoton wachsenden) $f : \mathbb{N} \rightarrow \mathbb{N}$ gilt:

$$\text{NTime}(f) \subseteq \text{DSpace}(f)$$

Korollar 4.8

$$\text{P} \subseteq \text{NP} \subseteq \text{PSpace}$$

Echtheit vermutet, aber unbewiesen (NP-versus-PSpace-Problem)

Zeit versus Platz (2)

Für Antwort Nr. 2 verwenden wir Konfigurationsgraphen.

Definition 4.9 (Konfigurationsgraph)

Sei M eine (1-Band-)NTM und $s \in \mathbb{N}$.

$\text{Conf}_{M,s}$ ist die Menge aller Konfigurationen von M der Länge $\leq s$.

Der s -Konfigurationsgraph für M ist der gerichtete Graph $G_{M,s} = (V, E)$ mit

- $V = \text{Conf}_{M,s}$
- $E = \{(C, C') \mid C \vdash_M C'\}$

Wir können o. B. d. A. **genau eine** akzeptierende Konfiguration annehmen:

- Es gibt laut Def. NTM sowieso nur einen akzeptierenden Zustand.
- Vor Anhalten kann TM alle Bänder löschen und Köpfe ganz nach links fahren.

Zeit versus Platz (2)

Akzeptanz = Erreichbarkeit im Konfigurationsgraphen:

Lemma 4.10

Sei M s -platzbeschränkte NTM mit akzeptierender Konfiguration C_{acc} und w Eingabe der Länge n .

Dann gilt: $w \in L(M)$ gdw. C_{acc} von $q_0 w$ aus in $G_{M,s(n)}$ erreichbar

Der Konfigurationsgraph ist **exponentiell groß**:

$$G_{M,s} \text{ hat Größe } |Q| \cdot |\Gamma|^s \cdot s \in 2^{\mathcal{O}(s)}$$

mögliche Zustände, Bandinhalte, Kopfpositionen

Zeit versus Platz (2)

Benötigen außerdem Platzkonstruierbarkeit:

Definition 4.11 (Platzkonstruierbarkeit)

Eine Funktion $s : \mathbb{N} \rightarrow \mathbb{N}$ heißt **platzkonstruierbar**, wenn es eine **terminierende** DTM M gibt, so dass für alle $w \in \Sigma^*$ gilt:

$$\text{space}_M(w) = s(|w|)$$

Betrachte Platzkonstruierbarkeit als „**technische Annahme**“, die von allen **natürlichen** Funktionen erfüllt wird, z. B.:

- n^i ist platzkonstruierbar, für alle $i \geq 1$
- 2^n ist platzkonstruierbar

Zeit versus Platz (2)

Nun Antwort Nr. 2:

„Was man in Platz f berechnen kann, kann man auch in **Zeit** $2^{\mathcal{O}(f)}$ berechnen, sogar für **nichtdeterministischen Platz und deterministische Zeit**.“

Theorem 4.12

Für alle **platzkonstruierbaren** Funktionen $f : \mathbb{N} \rightarrow \mathbb{N}$ gilt:

$$\text{NSpace}(f) \subseteq \text{DTime}(2^{\mathcal{O}(f)})$$

Beweis.

Sei $L \in \text{NSpace}(f)$ und M f -platzbeschränkte NTM mit $L(M) = L$.

Konstruieren $2^{\mathcal{O}(f)}$ -zeitbeschränkte DTM M' mit $L(M') = L$ wie folgt:

0. Eingabe w mit $|w| = n$
1. Generiere deterministisch den Konfigurationsgraphen $G_{M,f(n)}$ **T4.4**
2. Entscheide, ob die (eindeutig bestimmte) akzeptierende Konfiguration von $q_0 w$ aus erreichbar ist

Laufzeitanalyse:

T4.5

Theorem 4.12 (wiederholt)

Für alle **platzkonstruierbaren** Funktionen $f : \mathbb{N} \rightarrow \mathbb{N}$ gilt:

$$\text{NSpace}(f) \subseteq \text{DTime}(2^{O(f)})$$

Korollar 4.13

$$P \subseteq NP \subseteq \text{PSpace} \subseteq \text{ExpTime}$$

Echtheit vermutet, aber unbewiesen (**PSpace-versus-ExpTime-Problem**)

Auch für **deterministischen Platz** ist keine bessere Schranke bekannt.

- 4.1 Platzkomplexitätsklassen
- 4.2 Polynomieller Platzverbrauch (PSpace)
- 4.3 Zeitkomplexität versus Platzkomplexität
- NEXT** → 4.4 Der Satz von Savitch
- 4.5 PSpace-Härte und -Vollständigkeit
- 4.6 Logarithmischer Platz (LogSpace)

Der Satz von Savitch

Definition von PSpace ähnelt der von P (Platz statt Zeit)

Naheliegend: nichtdeterministisches Pendant (ähnlich NP)

Analog zu alternativer Charakterisierung von NP:

$$\text{NSpace} := \bigcup_{i \geq 1} \text{NSpace}(n^i)$$

Interessanter Gegensatz:

- „P versus NP“ ist wichtigstes offenes Problem der Informatik
- „PSpace versus NPSpace“ wurde 1970 von Walter Savitch **gelöst**: es gilt **PSpace = NPSpace!**

Theorem 4.14 (Savitch 1970)

Wenn s platzkonstruierbar ist, dann $\text{NSpace}(s) \subseteq \text{DSpace}(s^2)$.

Beweisidee: Erreichbarkeit im Konfig.-graphen, aber raffiniert (siehe Th1 2)

Kapitel 4

- 4.1 Platzkomplexitätsklassen
- 4.2 Polynomieller Platzverbrauch (PSpace)
- 4.3 Zeitkomplexität versus Platzkomplexität
- 4.4 Der Satz von Savitch
- NEXT** → 4.5 PSpace-Härte und -Vollständigkeit
- 4.6 Logarithmischer Platz (LogSpace)

PSpace-Vollständigkeit

Zur Erinnerung:

Theorem 4.15

$P \subseteq NP \subseteq PSpace \subseteq ExpTime$

Echtheit unbekannt!

Wie bei P vs NP:
man behilft sich mit dem Begriff der **Härte** und **Vollständigkeit**,
basierend auf Polynomialzeit-Reduktionen.

Einfache Beobachtung:

Lemma 4.16

PSpace ist abgeschlossen unter Polynomialzeit-Reduktionen:
Wenn $L' \leq_p L$ und $L \in PSpace$, dann $L' \in PSpace$.

PSpace-Vollständigkeit

Definition 4.17 (PSpace-Härte, PSpace-Vollständigkeit)

Problem L ist

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

Beobachtung 4.18

Wenn L PSpace-hart und

- $L \in P$, dann $P = PSpace$;
- $L \in NP$, dann $NP = PSpace$.

Theorem 4.19

QBF ist PSpace-vollständig.

(ohne Beweis)

PSpace-Vollständigkeit

Natürlichere PSpace-vollständige Probleme:

- Universalitätsproblem für NEAs
- Schnittleerheitsproblem für NEAs:
gegeben NEAs $\mathcal{A}, \mathcal{A}'$, ist $L(\mathcal{A}) \cap L(\mathcal{A}') = \emptyset$?
- SQL-Anfrageproblem
gegeben Instanz von relationaler Datenbank D , SQL-Anfrage q ,
Tupel t : ist t in der Antwort von q auf D enthalten?
- Viele Spielprobleme, z. B.
Brettspiele mit 2 Personen wie Dame (auf Feldern beliebiger Größe)

Ausblick

- 4.1 Platzkomplexitätsklassen
- 4.2 Polynomieller Platzverbrauch (PSpace)
- 4.3 Zeitkomplexität versus Platzkomplexität
- 4.4 Der Satz von Savitch
- 4.5 PSpace-Härte und -Vollständigkeit

NEXT →

- 4.6 Logarithmischer Platz (LogSpace)**
(weiter am 4.6.)