

FGI 1

Automaten, Formale Sprachen,
Berechenbarkeit

Christopher Habel & Matthias Jantzen

Kap 22 (Komplexität)

Matthias Jantzen

M. Jantzen, FGI-1, SoSe 2009: 1

Montag, 6. Juli 2009

1

Definition von Komplexität

Welcher Aufwand an Rechenzeit und Speicherplatz ist erforderlich, um ein gegebenes Problem zu lösen?

- Mindestens benötigter Rechenaufwand sind **untere** Schranken für ein Problem!

Beweise oft sehr schwer! Daher ging man in der Komplexitätstheorie dazu über, verschiedene Probleme miteinander zu vergleichen, um so wenigstens eine Aussage über deren **relative Komplexitäten** zu bekommen.

Komplexitätsklassen statt Chomsky-Hierarchie!

- **Obere Schranke** ist jeder bekannte Algorithmus.

Strukturelle Komplexitätstheorie

konkrete Komplexitätstheorie (Konstruktion konkreter Algorithmen)

Algorithmik (Analyse und Konstruktion guter Algorithmen)

M. Jantzen, FGI-1, SoSe 2009: 2

Montag, 6. Juli 2009

2

Chomsky Hierarchie

Typ 0 Sprachen = aufzählbare Mengen

Typ 1 = kontextsensitive Sprachen

Typ 2 = kontextfreie Sprachen

Typ 3 = reguläre Mengen

Typ-0
Grammatik,
Turingmaschine
(NTM/DTM)

monotone Grammatik,
linear beschränkter
Automat (LBA)

kontextfreie Grammatik
Kellerautomat (PDA)

reguläre Grammatik
endlicher Automat
(NFA/DFA)

M. Jantzen, FGI-1, SoSe 2009: 3

Montag, 6. Juli 2009

3

Rechnungen (Wiederholung)

Definition aus FGI Vorlesung:

Eine endliche Rechnung $k_1 \vdash k_2 \vdash \dots \vdash k_t$ heißt

Erfolgsrechnung, wenn $k_1 \in \{q_0\} \cdot \Gamma^*$ und $k_t \in \Gamma^* \cdot F \cdot \Gamma^*$ gilt.

Es ist also erlaubt, dass auf k_t eine weitere Konfiguration folgt,
die Rechnung also noch weitergehen könnte!

Für die DTM $A = (Z, \Sigma, \Gamma, \delta, q_0, \#, Z_{\text{end}})$ bezeichnet $L(A)$ die von A
akzeptierte Sprache:

$$L(A) := \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \exists q \in Z_{\text{end}} : q_0 w \vdash^* uqv\}$$

Auch hier könnte die Rechnung also noch weitergehen!

Eine (D)TM hält, wenn es bei der vorliegenden Bandinschrift keine
Folgekonfiguration gibt!

M. Jantzen, FGI-1, SoSe 2009:

Montag, 6. Juli 2009

4

Randbedingungen für Zeit-Schranken

Platzbedarf wird gemessen in benutzten Bandfeldern;
Zeitbedarf wird gemessen in Anzahl von Schritten!

- Es ist nützlich anzunehmen, dass eine TM ihre Eingabe stets vollständig liest und ein weiteres Feld vorrückt, um deren Ende festzustellen.
 - TM **arbeitet mit Zeitbeschränkung** $t(n)$, wenn die kleinste Zahl der in einer w akzeptierenden Rechnung benutzten Konfigurationswechsel höchstens $t(|w|)$ ist, und das Wort vollständig gelesen wurde. Man sagt, dass die *NTM* dann $\max\{n+1, t(n)\}$ -zeitbeschränkt ist.

Randbedingungen für Platz-Schranken

Platzbedarf wird gemessen in benutzten Bandfeldern;
Zeitbedarf wird gemessen in Anzahl von Schritten!

- Wenn eine *NTM* mit der Platzbeschränkung $s(n)$ arbeitet, dann sollte $s(n) \geq 1$ für jedes $n \in \mathbb{N}$ sein, denn die TM muss sich wenigstens ein Feld auf dem Arbeitsband ansehen!
 - Eine *NTM* **arbeitet mit Platzbeschränkung** $s(n)$, wenn die kleinste Zahl der in einer w akzeptierenden Rechnung besuchten Bandzellen höchstens $s(|w|)$ ist, die *NTM* somit $\max\{1, \lceil s(n) \rceil\}$ -platzbeschränkt ist.

Wie definiert man Komplexitätsklassen?

- Zusammenfassen ähnlicher Eingaben
 - über die Art des Problems (Semantik) ist in der Regel nur sehr schwer eine Aussage zu machen
 - Kategorisierung nur nach **Länge des Eingabewortes**.

Definition 68:

Seien $s : \mathbb{R} \rightarrow \mathbb{R}$, $t : \mathbb{R} \rightarrow \mathbb{R}$. Eine *NTM* T ist

- **t -zeitbeschränkt** gdw. sie für **jedes akzeptierte** Eingabewort der Länge n mit der Zeitbeschränkung $t(n)$ arbeitet.
- **s -platzbeschränkt** genau dann, wenn sie für **jedes akzeptierte** Eingabewort der Länge n mit der Platzbeschränkung $s(n)$ arbeitet.

wichtige Komplexitätsklassen

Definition 69:

\mathcal{P} ist die Klasse der Sprachen (algorithmischen Probleme), die man deterministisch in Polynomialzeit erkennen (lösen) kann.

$\mathcal{P}Space$ ist die Klasse der Sprachen (algorithmischen Probleme), die man mit polynomiell viel Platz erkennen (lösen) kann.

- Obige Klassen sind unabhängig vom verwendeten Modell (*DTM*, Registermaschine, Programmiersprache, ...).

Zusammenhang von Zeit- und Platzkomplexität

- Eine TM, die $t(n)$ Zeit (d.h. Schritte) zur Verfügung hat, kann nicht mehr als $t(n)$ Bandzellen besuchen.
 - Umgekehrt gilt dies nicht!
- Platz kann **wiederverwendet** werden, Zeit nicht!
- *Kann man trotzdem eine maximale Rechenzeit in Abhängigkeit vom verbrauchten Platz angeben?*
 - Ja! **Idee:** Sobald eine Konfiguration ein zweites Mal besucht wird, befinden wir uns in einer Endlosschleife!
 - Bleibt zu berechnen, **wieviele verschiedene Konfigurationen** bei einer gegebenen Platzbeschränkung möglich sind.

Anzahl der Konfigurationen

- Wovon hängt die Anzahl der möglichen unterschiedlichen Konfigurationen ab?
 - maximal verbrauchter Platz
 - * **Kopfpositionen** auf den Arbeitsbändern
 - * **Länge** möglicher Bandinschriften
 - Anzahl der Bandsymbole
 - * **Permutationen** von Bandsymbolen
 - Länge der Eingabe
 - * **Kopfpositionen** auf dem Eingabeband
 - Anzahl der Zustände in der endlichen Steuerung
 - * aktueller **Zustand** einer Konfiguration

Maximalzahl möglicher Konfigurationen

Eine Konfiguration hat die Schreibweise $k = uqv$.

Es gibt $|Q|$ viele Zustände und $|uv| + 1 = n + 1$ verschiedene Werte für die Position des Symbols q vor, hinter oder in der Bandinschrift uv . Ist maximal $s(n)$ Platz vorhanden, wird stets $|uv| \leq s(n)$ gelten.

in jedem Feld ein Symbol!

Es gibt maximal $|\Gamma|^{s(n)}$ viele verschiedene Inschriften.

Insgesamt kann die DTM M also in $s(n)$ Platz maximal

$$(n + 1) \cdot c_1^{s(n)} = 2^{\log_2(n+1)} c_1^{s(n)} = c_1^{\log_{c_1}(2) \log_2(n+1) + s(n)} = c_1^{c_2 \log_2(n+1) + s(n)}$$

viele Konfigurationen beim Akzeptieren durchlaufen.

Zeit versus Platz I

Offensichtlich gilt ja

$$\mathcal{DTime}(f(n)) \subseteq \mathcal{DSpace}(f(n)),$$

weil keine TM mehr Felder benutzen kann, als ihr Schritte zur Verfügung stehen!

Über den gerichteten Graphen aller Folgekonfigurationen, von denen maximal $s(n)$ viele zum Akzeptieren durchlaufen werden müssen erhalten wir auch für \mathcal{NSPACE} ein Ergebnis: Insgesamt gab es $c^{\log(n+1) + s(n)}$ viele Konfigurationen, die überprüft werden müssen (und auch können)! Es folgt so:

$$\mathcal{NSpace}(s(n)) \subseteq \mathcal{DTime}(k^{\log(n)+s(n)})$$

wichtige Folgerung

Azeptiert eine Turingmaschine eine Sprache L mit
einer Zeitbeschränkung $t(n)$
oder mit einer
Platzbeschränkung $s(n)$,
so ist L eine entscheidbare Menge!

Die Familien
 \mathcal{P} , \mathcal{NP} , $\mathcal{P}Space$, $\mathcal{NP}Space$
und ALLE über Komplexitätsbeschränkungen
(mit berechenbaren Funktionen) definierte
Sprachfamilien sind Teil der Familie der
entscheidbaren Sprachen!

M. Jantzen, FGI-1, SoSe 2009: 13

Montag, 6. Juli 2009

13

Zeit versus Platz II

Es gilt sogar:

$$\mathcal{N}Time(f(n)) \subseteq \mathcal{D}Space(f(n))$$

Beweisidee:

Sei M_1 eine k -NTM die L in $f(n)$ Zeit akzeptiert. DTM M_2 erzeugt eine der möglichen Konfigurationsfolgen der NTM M_1 als $f(n)$ lange Folge von natürlichen Zahlen aus $\{0,1,\dots,d\}$, den Adressen eines imaginären Konfigurationsbaumes (d ist Maximalzahl von Verzweigungen in einem Zustand). Das Überprüfen einer erzeugten Rechnung auf Akzeptanz benötigt nur $f(n)$ Zeit und Platz. Es müssen exponentiell viele Rechnungen überprüft werden, was durch erneute Verwendung des benutzten Platzes nicht mehr Platz braucht. Jede Zahlenfolge kann in immer wieder genutztem Platz notiert werden, der maximal $c \cdot f(n)$ viele Felder enthält! (vergl. Simulation von NTM durch DTM)

M. Jantzen, FGI-1, SoSe 2009: 14

Montag, 6. Juli 2009

14

Arten von Problemen

Je nach Aufgabenstellung lassen sich verschiedene Grundtypen von Problemen unterscheiden:

1. Entscheidungsprobleme
2. Suchprobleme
3. Optimierungsprobleme
4. Abzählungsprobleme
5. Anzahlprobleme

Notation von Problemen

Darstellung von Problemen grundsätzlich so:

Gegeben: \langle Angabe aller verwendbaren Eingabedaten. \rangle

Frage: \langle Problemstellung mit geforderter Ausgabe. \rangle

Antwort: \langle Gewünschte, erwartete Antwort(Möglichkeit)en. \rangle

Ein Problem beschreibt **eine Klasse von gleichartigen Fragestellungen**, nicht nur eine einzelne Frage:

NICHT:

„Ist die Formel $(x \vee (y \wedge \neg x))$ erfüllbar?“

SONDERN:

„Hat eine beliebig vorgegebene aussagenlogische Formel eine erfüllende Belegung?“

Probleminstanzen

In der Formulierung eines Problems kommen i.A. gewisse Parameter vor. Für die Bestimmung von

$$\text{ggT}(m, n)$$

sind dies $m, n \in \mathbb{N}$.

Werden alle vorkommenden Parameter durch konkrete Werte ersetzt, so erhält man eine konkrete Aufgabenstellung, eine **In-
stanz** des Problems.

Setzen wir in unserem Beispiel $m = 125$ und $n = 35$, so bedeutet

$$\text{ggT}(125, 35)$$

die konkrete Aufgabe, den größten gemeinsamen Teiler der beiden Zahlen 125 und 35, also 5, zu bestimmen

M. Jantzen, FGI-1, SoSe 2009: 17

Montag, 6. Juli 2009

17

Problem lösen *versus* Sprache akzeptieren

Problem **Prim?**

Gegeben: Die Binärdarstellung einer natürlichen Zahl $n \in \mathbb{N}$.
Gesucht: Ist n eine Primzahl?
Antwort: JA oder NEIN

Problem **Prim?** ist *Entscheidungsproblem*, d.h., Die Lösungen können nur aus der Menge { JA , NEIN } stammen.

Die zum Problem **Prim?** gehörende Sprache wäre:

$$L_{\text{prim}} := \left\{ w \in \{0, 1\}^* \mid [w]_2 \text{ ist Primzahl} \right\}$$

M. Jantzen, FGI-1, SoSe 2009: 18

Montag, 6. Juli 2009

18

Suchprobleme

... zum Beispiel:

Gegeben:	ungerichteter Graph $G := (V, E)$ mit $E \subseteq V \times V$ und Knoten $v_1, v_2 \in V$.
Gesucht:	ein Weg von v_1 nach v_2
Antwort:	Kantenfolge eines Pfades oder „Es gibt keinen!“

Optimierungsprobleme

... zum Beispiel:

Gegeben:	gerichteter, bewerteter Graph $G := (V, E)$ mit $E \subseteq V \times \mathbb{N} \times V$ und Knoten $v_1, v_2 \in V$.
Gesucht:	ein günstigster Weg von v_1 nach v_2
Antwort:	Kantenfolge eines Pfades (evtl. mit Kosten) oder „Es gibt keinen!“

Abzählprobleme

...zum Beispiel :

Gegeben:	endliche Menge von Objekten
Gesucht:	alle binären Suchbäume für diese Objekte
Antwort:	Aufzählung der binären Suchbäume

Gebiete konkreter Algorithmen und deren Programmierung
wird das Modul
„Algorithmen und Datenstrukturen (AD)“
(jedes Wintersemester)
behandeln!

M. Jantzen, FGI-1, SoSe 2009: 21

Montag, 6. Juli 2009

21

Notationsvarianten

In der Literatur wird bisweilen
 $TIME(t)$ anstelle von $DTIME(t)$
und
 $SPACE(s)$ anstelle von $DSPACE(s)$
notiert!

Unsere Schreibweise mit dem Anfangsbuchstaben in
Skript, ist in der Literatur eher selten.

M. Jantzen, FGI-1, SoSe 2009: 22

Montag, 6. Juli 2009

22

Beispiele

$$\{a^k b^k c^k \mid k \in \mathbb{N}\} \in \mathcal{DSpace}(\log n)$$

$$\{w\$w \mid w \in \Sigma^*\} \in \mathcal{DSpace}(\log n)$$

$$\{p \in \{1\}\{0, 1\}^* \mid [p]_2 \text{ ist Primzahl}\} \in \mathcal{DSpace}(n)$$

$$\{p \in \{1\}\{0, 1\}^* \mid [p]_2 \text{ ist Primzahl}\} \in \mathcal{DTime}(n^{12})$$

Letzteres Ergebnis vom 10. Juli 2002 stammt von Manindra Agrawal und den Bachelor-Studenten Neeraj Kayal und Nitin Saxena!

Wie zeigen *wir/Sie* die ersten drei Aussagen?

M. Jantzen, FGI-1, SoSe 2009: 23

Montag, 6. Juli 2009

23

Speed up und Kompression

Satz 45:

Speed up

Zu jeder $t(n)$ -zeitbeschränkten TM mit $\inf_{n \rightarrow \infty} \left(\frac{t(n)}{n}\right) = \infty$ und jedem $c \in \mathbb{R}$ mit $c > 0$ gibt es eine äquivalente $c \cdot t(n)$ -zeitbeschränkte TM.

Satz 46:

Kompression

Zu jeder $s(n)$ -platzbeschränkten TM und jeder reellen Zahl $c \in \mathbb{R}$ mit $c > 0$ gibt es eine äquivalente TM, die $c \cdot s(n)$ -platzbeschränkt ist.

Konstruktionen
ähnlich der
LBA-Konstruktion
zu einer
kontextsensitiven
Grammatik!
(Blockbildung zu
neuen Symbolen)

M. Jantzen, FGI-1, SoSe 2009: 24

Montag, 6. Juli 2009

24

grundlegende Komplexitätsklassen III

Wegen der **Kompressions-** und **Speed Up-** Sätze folgt:

$$\mathcal{P} = \bigcup_{i \geq 1} \mathcal{D}Time(n^i) = \bigcup_{p \in POLY} \mathcal{D}Time(p(n))$$

$$\mathcal{NP} = \bigcup_{i \geq 1} \mathcal{N}Time(n^i) = \bigcup_{p \in POLY} \mathcal{N}Time(p(n))$$

$$\mathcal{P}Space = \bigcup_{i \geq 1} \mathcal{D}Space(n^i) = \bigcup_{p \in POLY} \mathcal{D}Space(p(n))$$

$$\mathcal{NP}Space = \bigcup_{i \geq 1} \mathcal{N}Space(n^i) = \bigcup_{p \in POLY} \mathcal{N}Space(p(n))$$

Hierbei bezeichnet $POLY$ die Klasse aller Polynome aus $\mathbb{R}[x]$!

M. Jantzen, FGI-1, SoSe 2009: 25

Montag, 6. Juli 2009

25

einfache Inklusionen

Diese Inklusionen folgen aus den Definitionen und der Beobachtung, dass in $f(n)$ Zeit nicht mehr als $f(n)$ Platz verbraucht werden kann!

$$\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{P}Space \subseteq \mathcal{NP}Space$$

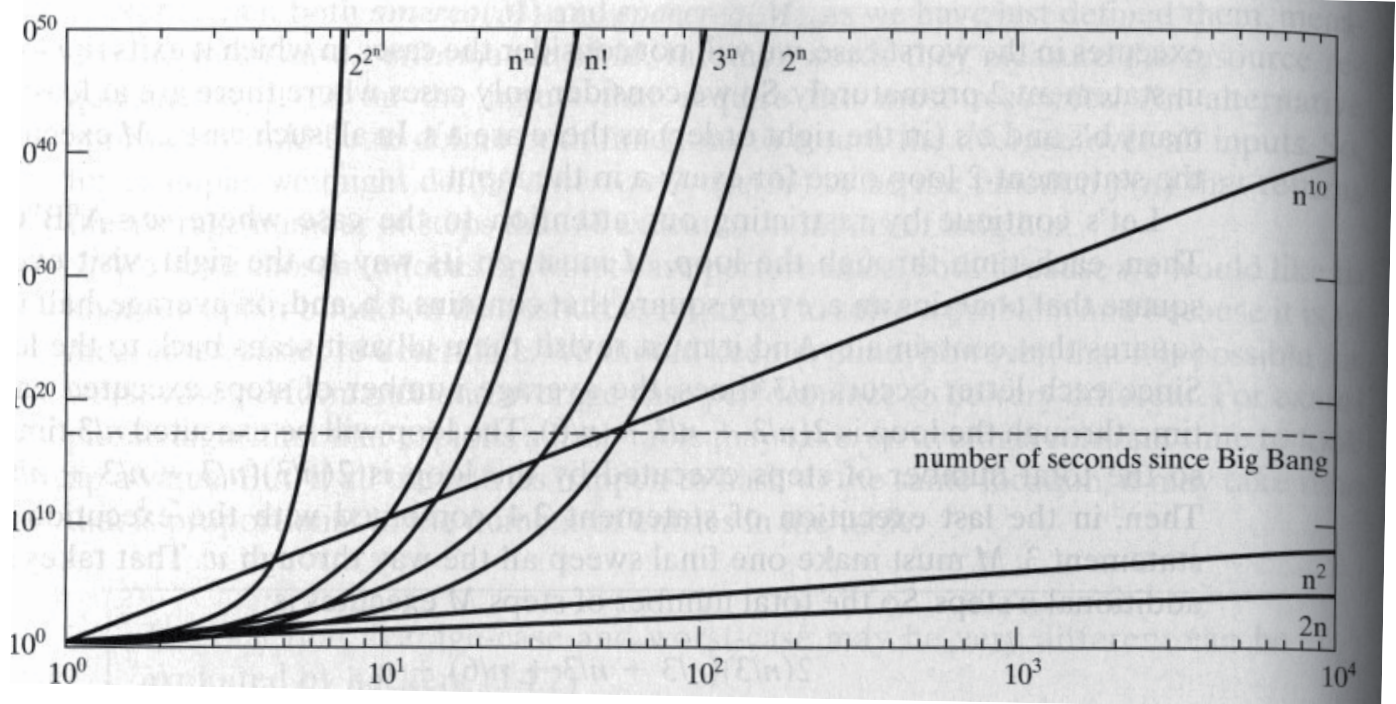
Wegen der Kompressions- und Speed-up Theoreme, sind **konstante Faktoren keine wirkliche Unterscheidung** zwischen den Funktionen. Daher werden wir Funktionen nach der „**Größenordnung**“ klassifizieren!

M. Jantzen, FGI-1, SoSe 2009: 26

Montag, 6. Juli 2009

26

Wachstumsvergleich



M. Jantzen, FGI-1, SoSe 2009: 27

Montag, 6. Juli 2009

27

Landau Notation

Definition 72:

Eine Funktion $f : \mathbb{N} \rightarrow \mathbb{R}$ wächst mit der Ordnung $g(n)$ bzw. g , notiert durch $f(n) \in O(g(n))$, falls

$g : \mathbb{N} \rightarrow \mathbb{R}$ eine Funktion ist und eine Konstante $c \in \mathbb{R}^+ := \{x \in \mathbb{R} \mid x > 0\}$ existiert, so dass $|f(n)| \leq c \cdot |g(n)|$ für alle, bis auf endlich viele $n \in \mathbb{N}$ gilt.

Etwas knapper notiert liest sich das so:

$$O(g(n)) = \left\{ f : \mathbb{N} \rightarrow \mathbb{R} \mid (\exists c \in \mathbb{R}^+) (\exists n_0 \in \mathbb{N}) (\forall n \geq n_0) [|f(n)| \leq c |g(n)|] \right\}.$$

Kurz: f wächst nicht schneller als g !

M. Jantzen, FGI-1, SoSe 2009: 28

Montag, 6. Juli 2009

28

Beispiel

Als weitere Beispiele seien f und g definiert durch:

$$f(n) := 12n^4 - 11n^3 + 1993 \text{ und } g(n) := 7n^3 - n.$$

Dann ist $g \in O(f)$ aber nicht $f \in O(g)$.

Auch gilt $f \in O(n^4)$ oder $f \in O(12000n^4 - 30000)$.

Andere Schreibweisen für $f \in O(g)$ sind in der Literatur auch $f = O(g)$ oder sehr selten $f \simeq O(g)$.

Rechenregeln

$$n^m \in O(n^{m'}) \text{ falls } m \leq m' \quad (1)$$

$$f(n) \in O(f(n)) \quad (2)$$

$$cO(f(n)) = O(f(n)) \quad (3)$$

$$O(O(f(n))) = O(f(n)) \quad (4)$$

$$O(f(n)) + O(g(n)) = O(|f(n)| + |g(n)|) \quad (5)$$

$$O(f(n))O(g(n)) = O(f(n)g(n)) \quad (6)$$

$$O(f(n)g(n)) = f(n)O(g(n)) \quad (7)$$

Definition 73:

Wir definieren neben $O(f(n))$ noch weitere asymptotische Funktionenklassen:

$o(g(n))$ ("klein oh"),

$\Omega(g(n))$ ("groß Omega"),

$\omega(g(n))$ ("klein Omega") und

$\Theta(g(n))$ ("Theta"),

was in diesem Zusammenhang nur als Großbuchstabe auftritt),

und sagen dann:

eine Funktion $f(n)$ ist **von der Ordnung** $\mathcal{O}(g(n))$, wenn sie für $\mathcal{O} \in \{O, o, \Omega, \omega, \Theta\}$ zur Menge $\mathcal{O}(g(n))$ gehört.

klein ,o‘ etc.

noch zu Def. 73:

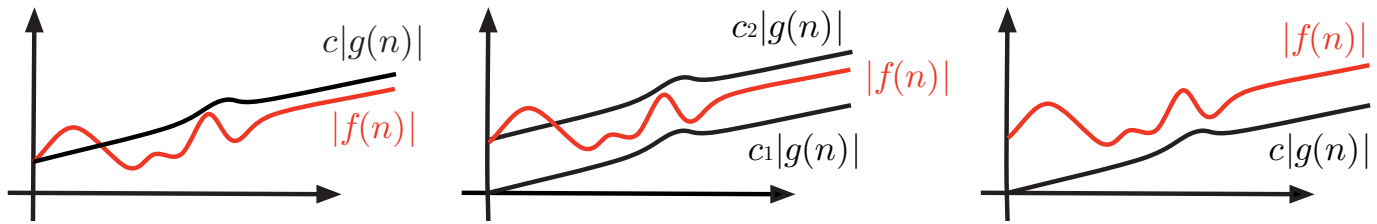
$$f(n) \in o(g(n)) \text{ ist äquivalent zu } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) \in \omega(g(n)) \iff g(n) \in o(f(n))$$

$$f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$$

$$f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n)) \text{ und } f(n) \in \Omega(g(n)).$$

Θ einmal bildlich



$f \in \Theta(g)$ statt umständlicher $f(n) \in \Theta(g(n))$

eine Vergleichstabelle

A	B	O	o	Ω	ω	Θ
$\lg^k n$	n^ϵ	×	×			
n^k	c^n	×	×			
\sqrt{n}	$n^{\sin n}$					
2^n	$2^{n/2}$			×	×	
$n^{\lg \epsilon}$	$\epsilon^{\lg n}$	×		×		×
$\lg(n!)$	$\lg(n^n)$	×		×		×

Komplexitätstheorie

