
16. Allgemeine Komplexitätsmaße

Bei der Komplexität von Algorithmen (d.h. Programmen) unterscheidet man zwischen *statischer* und *dynamischer* Komplexität, die als Kostenfunktionen für die Entwicklung bzw. den Betrieb von Programmen dienen. Die statische Komplexität, auf die wir hier nicht weiter eingehen werden, bezieht sich auf das Programm selbst als Text (Syntax). Ein mögliches einfaches Komplexitätsmaß ist hier die Länge des Programms.¹ Die dynamische Komplexität betrifft dagegen den Programmablauf (Semantik). Die wichtigsten Komplexitätsmaße hier sind Laufzeit und Speicherbedarf eines Programms auf einer zugehörigen Maschine. Zur Beschreibung der (dynamischen) Komplexität einer berechenbaren Funktion f (oder einer entscheidbaren Menge M) betrachtet man die Algorithmen, die f (bzw. c_M) berechnen. Diese liefern *obere* und *untere* Komplexitätsschranken:

- o ist eine obere Schranke für f , falls es ein Programm P gibt, das f berechnet und dessen Komplexität nach oben durch o beschränkt ist.
- u ist eine untere Schranke für f , falls für jedes Programm P , das f berechnet, dessen Komplexität nach unten durch u beschränkt ist.

Die Komplexität von f liegt dann in dem Intervall $[u, o]$. Offensichtlich gilt $u \leq o$ für jede untere Schranke u und jede obere Schranke o von f . Um die Komplexität von f möglichst exakt zu bestimmen, versucht man obere und untere Schranken u und o anzugeben, die möglichst nahe beieinander liegen. Man kann jedoch i. Allg. $u = o$ nicht erreichen (siehe den Beschleunigungssatz weiter unten). Für viele interessante Funktionen f sind die Lücken zwischen bekannten unteren und oberen Schranken sehr groß.

In diesem Abschnitt wollen wir den Begriff eines Komplexitätsmaßes axiomatisch einführen und damit Eigenschaften aller Komplexitätsmaße beweisen. In den folgenden Abschnitten werden wir uns dann auf die wichtigsten konkreten Komplexitätsmaße - Rechenzeit und Speicherbedarf - für Turingmaschinen beschränken.

Intuitiv ordnet ein *Komplexitätsmaß* jedem Programm P (eines festen Programmiersystems S), das eine n -stellige partielle Funktion φ_P berechnet, eine *Kosten-* oder *Schrittzahlfunktion* Φ_P zu, die für jede Eingabe \vec{x} die Kosten $\Phi_P(\vec{x})$ der Berechnung von $\varphi_P(\vec{x})$ mit Hilfe von P angibt. Dabei gehen wir davon aus, dass die Kosten einer terminierenden Rechnung eine natürliche Zahl und die Kosten einer divergierenden Rechnung unendlich sind. Identifizieren wir $\Phi_P(\vec{x}) \uparrow$ mit $\Phi_P(\vec{x}) = \infty$, so kann Φ_P also als partielle Funktion mit Werten aus \mathbb{N} aufgefasst werden, wobei der Definitionsbereich von Φ_P mit dem von φ_P übereinstimmt. Weiter ergeben sich (bei einer sinnvollen

¹Ein differenzierteres statisches Kostenmaß, das auch die „Schwierigkeit“ der Programme berücksichtigt, ist die *Kolmogorov-Komplexität*. Die Idee ist hier, statt der Länge des Programmtextes T selbst die Länge der kürzesten Beschreibung (Darstellung) von T zu wählen. Sinnvoll ist dies natürlich nur bezüglich einer fest vorgegebenen Darstellungsweise. Bei der Kolmogorov-Komplexität wählt man hierzu ein universelles Programmiersystem S (Gödelnummerierung) und definiert die Komplexität von T als die Länge des kürzesten S -Programms, das bei leerer Eingabe den Programmtext T als Ausgabe liefert.

Kostenfunktion) die Kosten effektiv aus dem Programmablauf, d.h. die Kostenfunktion Φ_P ist partiell berechenbar und man kann effektiv feststellen, ob die Kosten einen vorgegebenen Wert y annehmen. Die Kosten wachsen nämlich mit der Rechnung und man kann zu jedem Zeitpunkt der Rechnung die bislang angefallenen Kosten effektiv feststellen. Um zu entscheiden, ob $\Phi_e(x) = y$ gilt, simuliert man also die Berechnung von $\varphi_e(x)$ bis diese entweder terminiert (und man die Endkosten $\Phi_e(x)$ ablesen kann), oder die aktuellen Kosten y überschreitet (und damit $\Phi_e(x) > y$ gilt). Da für $\varphi_e(x) \uparrow$ gilt, dass $\Phi_e(x) = \infty$ ist, muss eines dieser beiden Ergebnisse eintreten.

Beschränken wir uns auf den Fall 1-stelliger, berechenbarer Funktionen über \mathbb{N} , so können wir obige intuitive Charakterisierung von Komplexitätsmaßen mit Hilfe der Church-Turing-These wie folgt formalisieren. Dabei muss man nur beachten, dass man aus jedem „natürlichen“ universellen Programmiersystem durch Gödelisierung Gödelnummerierungen der partiell berechenbaren Funktionen erhält.

16.1 DEFINITION. Ein *allgemeines Komplexitätsmaß* (AKM) der einstelligen partiell rekursiven Funktionen ist ein Paar (φ, Φ) bestehend aus einer Gödelnummerierung φ der 1-stelligen partiell rekursiven Funktionen und einer 2-stelligen partiell rekursiven Funktion Φ mit den beiden folgenden Eigenschaften:

$$(A1) \quad Db(\varphi) = Db(\Phi)$$

$$(A2) \quad \text{Graph}(\Phi) = \{(e, x, y) : \Phi_e(x) = y\} \text{ ist rekursiv.}$$

Man nennt den e -ten Zweig Φ_e die *Kosten-* bzw. *Schrittzahlfunktion* von φ_e bzgl. (φ, Φ) .

Das zweite Axiom (A2) impliziert, dass die üblichen Vergleichsprädikate für die Kosten, wie z.B.

$$P(e, x, y) \Leftrightarrow \Phi_e(x) \leq y$$

oder

$$P'(e, x, y) \Leftrightarrow \Phi_e(x) > y$$

rekursiv sind. Da

$$P(e, x, y) \Leftrightarrow \exists z \leq y ((e, x, z) \in \text{Graph}(\Phi))$$

und

$$P'(e, x, y) \Leftrightarrow \neg P(e, x, y + 1)$$

folgt dies nämlich aus der Rekursivität von $\text{Graph}(\Phi)$ mit Hilfe der früher gezeigten Abschlusseigenschaften der Klasse der rekursiven Mengen.

16.2 BEISPIEL. Beispiele für AKM sind die Zeit- und Platzkomplexität von Turingmaschinen. In Abschnitt 11 haben wir schon gezeigt, dass man durch Gödelisierung der normierten 1-Band-Turingmaschinen eine Gödelnummerierung φ von $F^{(1)}$ (REK) erhält. Entsprechend erhält man eine Gödelnummerierung φ durch Gödelisierung aller 1-Band-Turingmaschinen oder aller k -Band-Turingmaschinen ($k \geq 2$ fest) oder aller

Mehrband-Turingmaschinen. Sei im Folgenden solch eine Gödelnummerierung φ festgehalten und M_e die e -te entsprechende Turingmaschine, d.h. M_e berechnet φ_e . Dann ist (φ, T) ein AKM, wobei

$$T(e, x) = \begin{cases} \text{time}_{M_e}(x) & \text{falls } M_e \text{ bei Eingabe } x \text{ terminiert} \\ \uparrow & \text{sonst.} \end{cases}$$

Hierbei ist $\text{time}_{M_e}(x)$ die schon früher definierte Rechenzeit von M_e bei Eingabe x , d.h. die Länge der Rechnung der Maschine M_e bei dieser Eingabe. Entsprechend ist (φ, S) ein AKM, wobei

$$S(e, x) = \begin{cases} \text{space}_{M_e}(x) & \text{falls } M_e \text{ terminiert bei Eingabe } x \\ \uparrow & \text{sonst.} \end{cases}$$

Hierbei ist $\text{space}_{M_e}(x)$ der Platzbedarf von M_e bei Eingabe x . D.h. bei einer 1-Band-Turingmaschine ist $\text{space}_{M_e}(x)$ die Anzahl der Felder, die im Verlauf der Rechnung Arbeitsfeld sind. Bei einer k -Band-Turingmaschine ($k \geq 2$) definiert man den Platzbedarf $\text{space}_{M_e}^i(x)$ für jedes Band i entsprechend und nimmt das Maximum hiervon als gesamten Platzbedarf ($\text{space}_{M_e}(x) = \max\{\text{space}_{M_e}^i(x) : 1 \leq i \leq k\}$.)

Dass (φ, T) und (φ, S) AKM sind, sieht man leicht. Im Fall von (φ, S) muss man sich zum Nachweis von (A2) nur klar machen, dass für gegebene Turingmaschinen M , Eingabe x und Kostenschwelle y einer der folgenden drei Fälle nach endlich vielen Rechenschritten auftritt, und aus dem auftretenden Fall „ $\Phi_e(x) = y?$ “ entschieden werden kann:

1. M stoppt ohne zuvor mehr als y Felder auf einem Band besucht zu haben.
2. M besucht erstmals das $(y + 1)$ -te Feld auf einem Band.
3. Eine Konfiguration von M wiederholt sich, bevor M mehr als y Felder auf einem Band besucht hat (Endlosschleife, d.h. M divergiert.)

16.3 BEISPIEL. Pathologische Beispiele für AKM kann man durch Variation eines gegebenen AKM (φ, Φ) erhalten. So gibt es zu jeder total rekursiven Funktion $f^{(1)}$ ein AKM $(\hat{\varphi}, \hat{\Phi})$, in dem f zum Nulltarif berechnet werden kann. Hierzu wählt man einen φ -Index e_0 für f , setzt $\hat{\varphi} = \varphi$ und

$$\hat{\Phi}(e, x) = \begin{cases} \Phi(e, x) & \text{falls } e \neq e_0 \\ 0 & \text{sonst.} \end{cases}$$

Um die Komplexität von Funktionen $f^{(1)} \in \text{F(REK)}$ bezüglich eines AKM (φ, Φ) zu beschreiben, fassen wir alle Funktionen mit einer vorgegebenen oberen Schranke t in einer Komplexitätsklasse zusammen. Entsprechend verfahren wir für rekursive Mengen $A \subseteq \mathbb{N}$, indem wir die Komplexität der Menge A mit der Komplexität der charakteristischen Funktion c_A von A gleichsetzen.

16.4 DEFINITION. Sei (φ, Φ) ein allgemeines Komplexitätsmaß, und sei $t : \mathbb{N} \rightarrow \mathbb{N}$ total rekursiv. Eine totale Funktion $f : \mathbb{N} \rightarrow \mathbb{N}$ heißt t -beschränkt bezüglich (φ, Φ) , falls es ein $e \in \mathbb{N}$ mit $f = \varphi_e$ gibt, sodass

$$\forall x (\Phi_e(x) \leq t(x)).$$

Die Klasse

$$FC_{(\varphi, \Phi)}(t) = \{f : f t\text{-beschränkt bzgl. } (\varphi, \Phi)\}$$

heißt die (φ, Φ) -Komplexitätsklasse mit Schranke t . Entsprechend heißt $A \subseteq \mathbb{N}$ t -beschränkt bzgl. (φ, Φ) , falls c_A t -beschränkt bzgl. (φ, Φ) ist und man definiert

$$C_{(\varphi, \Phi)}(t) = \{A \subseteq \mathbb{N} : A t\text{-beschränkt bzgl. } (\varphi, \Phi)\}.$$

Im Zentrum der Komplexitätstheorie steht der Vergleich von Komplexitätsklassen mit unterschiedlichen Schranken und/oder bezüglich unterschiedlicher Komplexitätsmaße. Hierbei beschränkt man sich häufig auf die Komplexitätsklassen von Mengen. Trivialerweise gilt für jedes AKM (φ, Φ)

$$t \leq t' \text{ fast überall} \Rightarrow C_{(\varphi, \Phi)}(t) \subseteq C_{(\varphi, \Phi)}(t') \quad (16.1)$$

$$\text{REK} = \bigcup_{t \text{ tot.rek.}} C_{(\varphi, \Phi)}(t) \quad (16.2)$$

Tieferliegende Fragen hängen i. Allg. von dem AKM (φ, Φ) ab, einige Phänomene lassen sich jedoch für alle AKM nachweisen. So können wir z.B. zeigen, dass (für beliebiges AKM) keine Komplexitätsklasse $C_{(\varphi, \Phi)}(t)$ alle rekursiven Mengen enthält.

16.5 SATZ. Sei (φ, Φ) ein AKM und sei $t : \mathbb{N} \rightarrow \mathbb{N}$ total rekursiv. Dann gilt $C_{(\varphi, \Phi)}(t) \subsetneq \text{REK}$.

BEWEIS. Es genügt wegen (16.2), eine total rekursive, 0-1-wertige Funktion $f^{(1)}$ anzugeben, die bzgl. (φ, Φ) nicht t -beschränkt ist. Hierzu definieren wir

$$f(\langle e, x \rangle) = \begin{cases} 1 - \varphi_e(\langle e, x \rangle) & \text{falls } \Phi_e(\langle e, x \rangle) \leq t(\langle e, x \rangle) \\ 0 & \text{sonst.} \end{cases}$$

Wegen (A2) ist f (partiell) rekursiv und wegen (A1) ist f total. Weiter gilt für jeden φ -Index e von f (d.h. $f = \varphi_e$), dass $\Phi_e(\langle e, x \rangle) > t(\langle e, x \rangle)$ für alle $x \in \mathbb{N}$ (da andernfalls $f(\langle e, x \rangle) \neq \varphi_e(\langle e, x \rangle)$ nach Definition von f). Also ist f nicht t -beschränkt bzgl. (φ, Φ) . \square

Nach Satz 16.5 gibt es (wegen (16.2)) zu jeder rekursiven Funktion t eine rekursive Funktion t' mit $C_{(\varphi, \Phi)}(t) \subsetneq C_{(\varphi, \Phi)}(t')$. Dies führt zu der Frage, wie man zu gegebenem t ein solches t' passend wählen kann. Wie der nächste Satz zeigt, reicht es hierzu i. Allg. nicht, t mit Hilfe einer vorgegebenen rekursiven Funktion g zu vergrößern.

16.6 SATZ. (LÜCKENSATZ) Sei (φ, Φ) ein allgemeines Komplexitätsmaß und seien $f^{(1)}$ und $g^{(1)}$ total rekursive Funktionen. Dann gibt es eine total rekursive Funktion t mit $f(x) \leq t(x)$ für alle x und

$$(\forall e)[\forall x(\Phi_e(x) \leq g(t(x))) \Rightarrow \forall x(\Phi_e(x) \leq t(x))] \quad (16.3)$$

D.h. $C_{(\varphi, \Phi)}(g(t(x))) \subseteq C_{(\varphi, \Phi)}(t(x))$.

BEWEIS. O.B.d.A können wir davon ausgehen, dass die Funktionen f und g streng monoton sind. Zur Definition von t definieren wir zunächst ein 2-stelliges Prädikat P durch

$$P(x, y) := y \geq f(x) \ \& \ \neg \exists e \leq x (y \leq \Phi_e(x) \leq g(y))$$

und setzen dann

$$t(x) := \mu y (P(x, y))$$

Wegen (A2) ist P rekursiv, also t partiell rekursiv. Weiter ist t total, da es zu jeder Zahl x ein y mit $P(x, y)$ gibt. (Hierzu beobachtet man, dass

$$E_x = \{\Phi_e(x) : e \leq x \ \& \ \Phi_e(x) \downarrow\} \cup \{f(x)\}$$

endlich ist und $P(x, y)$ für alle y mit $y > \max(E_x)$ gilt.) Die Funktion t ist aber gerade so gewählt, dass für alle x und $e \leq x$ der Wert $\Phi_e(x)$ nicht zwischen $t(x)$ und $g(t(x))$ liegt. Offensichtlich folgt hieraus (16.3). \square

Für den Vergleich von Komplexitätsmaßen ist die Beobachtung hilfreich, dass sich die Kosten $\Phi_e(x)$ zur Berechnung von $\varphi_e(x)$ in einem AKM (φ, Φ) rekursiv in der Eingabe x und den Kosten $\Psi_{h(e)}(x)$ zur Berechnung der φ_e entsprechenden Funktion $\Psi_{e'}(x)$ in einem beliebigen anderen AKM (ψ, Ψ) nach oben abschätzen lassen.

16.7 SATZ. (VERGLEICHBARKEITSSATZ) Seien (φ, Φ) und (ψ, Ψ) allgemeine Komplexitätsmaße und sei h eine rekursive Übersetzungsfunktion von φ nach ψ . Es gibt eine total rekursive streng monotone Funktion $g^{(2)}$ mit

$$\forall e \in \mathbb{N} \forall x \in \mathbb{N} (\Psi_{h(e)}(x) \leq g(x, \Phi_e(x)) \ \& \ \Phi_e(x) \leq g(x, \Psi_{h(e)}(x))). \quad (16.4)$$

BEWEIS. Wir definieren zunächst

$$\tilde{g}(e, x, y) := \begin{cases} \Phi_e(x) + \Psi_{h(e)}(x) & \text{falls } \Phi_e(x) = y \text{ oder } \Psi_{h(e)}(x) = y \\ 0 & \text{sonst.} \end{cases}$$

Wegen der Eigenschaft (A2) von Φ und Ψ ist \tilde{g} partiell rekursiv. Weiter ist \tilde{g} total, da

$$\begin{aligned} \Phi_e(x) \downarrow &\Leftrightarrow \varphi_e(x) \downarrow && ((A1) \text{ für } (\varphi, \Phi)) \\ &\Leftrightarrow \Psi_{h(e)}(x) \downarrow && (h \text{ Übersetzungsfunktion}) \\ &\Leftrightarrow \Psi_{h(e)}(x) \downarrow && ((A1) \text{ für } (\psi, \Psi)). \end{aligned}$$

Also ist auch

$$g(x, y) := \max\{\tilde{g}(e, x', y') : e \leq x \ \& \ x' \leq x \ \& \ y' \leq y\} + x + y$$

total rekursiv und nach Definition streng monoton in x und y . Zum Nachweis von (16.4) beobachtet man, dass für jedes e und jedes $x \geq e$ gilt:

$$\begin{aligned} \Psi_{h(e)}(x) &\leq \Phi_e(x) + \Psi_{h(e)}(x) \\ &= \tilde{g}(e, x, \Phi_e(x)) \\ &\leq g(x, \Phi_e(x)) \end{aligned}$$

und analog

$$\begin{aligned}\Phi_e(x) &\leq \Phi_e(x) + \Psi_{h(e)}(x) \\ &= \tilde{g}(e, x, \Psi_{h(e)}(x)) \\ &\leq g(x, \Psi_{h(e)}(x)).\end{aligned}$$

□

Der Vergleichbarkeitssatz für Komplexitätsmaße liefert den folgenden Vergleichbarkeitssatz für Komplexitätsklassen bezüglich verschiedener Komplexitätsmaße:

16.8 KOROLLAR. *Seien (φ, Φ) und (ψ, Ψ) allgemeine Komplexitätsmaße. Dann gibt es eine streng monotone rekursive Funktion $\hat{g}^{(1)}$, sodass für alle rekursiven Funktionen t mit $t(n) \geq n$ stets gilt:*

$$C_{(\varphi, \Phi)}(t) \subseteq C_{(\psi, \Psi)}(\hat{g}(t)).$$

BEWEIS. Sei h eine Übersetzungsfunktion von φ nach ψ und sei g die streng monotone rekursive Funktion aus dem Vergleichbarkeitssatz, die (16.4) erfüllt. Definiere \hat{g} durch

$$\hat{g}(x) := \max_{e \leq x} g(e, x).$$

Dann ist \hat{g} total rekursiv und streng monoton. Sei nun t total rekursiv, wobei $t(n) \geq n$ für alle n gelte, und sei $A \in C_{(\varphi, \Phi)}(t)$. Dann gibt es einen φ -Index e für c_A mit $\Phi_e(x) \leq t(x)$ fast überall. Da h Übersetzungsfunktion von φ nach ψ ist, gilt dann $c_A = \Psi_{h(e)}$ und

$$\begin{aligned}\Psi_{h(e)}(x) &\leq g(x, \Phi_e(x)) && ((16.4)) \\ &\leq g(x, t(x)) && (\Phi_e(x) \leq t(x), g \text{ monoton}) \\ &\leq g(t(x), t(x)) && (x \leq t(x), g \text{ monoton}) \\ &\leq \hat{g}(t(x)) && (\text{Definition von } \hat{g})\end{aligned}$$

für fast alle x , weshalb $A \in C_{(\psi, \Psi)}(\hat{g}(t))$. □

Mit dem Vergleichbarkeitssatz lassen sich gewisse Eigenschaften eines speziellen Komplexitätsmaßes auf alle Komplexitätsmaße übertragen. Wir geben ein Beispiel:

16.9 KOROLLAR. *Sei (ψ, Ψ) ein allgemeines Komplexitätsmaß und seien $f^{(1)}, t^{(1)}$ total rekursive Funktionen. Dann gibt es ein $e \in \mathbb{N}$ mit $\psi_e = f$ und $\Psi_e(x) \geq t(x)$ für fast alle x . (D.h. man kann für jede Funktion beliebig langsame Programme bzgl. jedes gegebenen AKM finden.)*

BEWEIS. Sei $(\varphi, \Phi) = (\varphi, T)$ das Zeitkomplexitätsmaß für Mehrband-Turingmaschinen, sei h eine Übersetzungsfunktion von φ nach ψ , und sei nach dem Vergleichbarkeitssatz g eine streng monotone rekursive Funktion, die (16.4) erfüllt. Weiter gelte o.B.d.A., dass t streng monoton ist.

Sei nun M eine Mehrband-Turingmaschine, die t berechnet, aber vor der eigentlichen Rechnung bei Eingabe x zunächst $\geq g(x, t(x))$ unnötige Schritte ausführt (z.B. auf einem später nicht benutzten Band $g(x, t(x))$ in Unärdarstellung produziert). Für

eine Gödelnummer e_0 von M gilt dann $\varphi_{e_0} = f$ und $\Phi_{e_0}(x) \geq g(x, t(x))$ für alle x . Für $e = h(e_0)$ gilt dann $\psi_e = f$ und wegen (16.4)

$$g(x, t(x)) \leq \Phi_{e_0}(x) \leq g(x, \Psi_e(x)) \text{ (für fast alle } x\text{).}$$

Da g streng monoton ist, impliziert dies $t(x) \leq \Psi_e(x)$ für fast alle x . \square

Aus Lücken- und Vergleichbarkeitssatz erhält man folgendes Kollabierungsergebnis für Komplexitätsklassen bzgl. verschiedener Komplexitätsmaße.

16.10 KOROLLAR. *Seien (φ, Φ) und (ψ, Ψ) allgemeine Komplexitätsmaße, sodass $C_{(\psi, \Psi)}(t) \subseteq C_{(\varphi, \Phi)}(t)$ für alle rekursiven Schranken t gilt². Dann gibt es zu jeder rekursiven Funktion t_0 eine rekursive Funktion $t_1 \geq t_0$ mit $C_{(\psi, \Psi)}(t_1) = C_{(\varphi, \Phi)}(t_1)$.*

BEWEIS. Es genügt $t_1 \geq t_0$ mit $C_{(\varphi, \Phi)}(t_1) \subseteq C_{(\psi, \Psi)}(t_1)$ zu finden. Nach Korollar 16.8 zum Vergleichbarkeitssatz gibt es ein rekursives \hat{g} mit $C_{(\varphi, \Phi)}(t) \subseteq C_{(\psi, \Psi)}(\hat{g} \circ t)$ für alle t mit $t \geq t_0$, da wir o.B.d.A. $t_0(n) \geq n$ annehmen können. Wenden wir nun den Lückensatz auf das AKM (ψ, Ψ) , die untere Schranke t_0 und die Lückengröße g an, so erhalten wir ein $t_1 \geq t_0$ mit $C_{(\psi, \Psi)}(\hat{g}(t_1)) = C_{(\psi, \Psi)}(t_1)$ und damit $C_{(\varphi, \Phi)}(t_1) \subseteq C_{(\psi, \Psi)}(t_1)$. \square

Aus Korollar 16.10 ergeben sich einige interessante Beobachtungen über die Beziehungen zwischen verschiedenen Komplexitätsmaßen. Wir geben zwei Beispiele:

16.11 BEISPIEL. Seien M_1 und M_2 universelle Computer, wobei M_2 schneller ist als M_1 , und seien (φ, Φ) und (ψ, Ψ) die zu M_1 bzw. M_2 gehörenden Zeitkomplexitätsmaße. Dann gilt stets $C_{(\varphi, \Phi)}(t) \subseteq C_{(\psi, \Psi)}(t)$ (da M_2 schneller als M_1 ist). Nach Korollar 16.10 gibt es also zu jedem t ein $t' \geq t$, sodass die von M_1 bzw. M_2 in der Zeit t' berechneten Funktionen übereinstimmen.

16.12 BEISPIEL. Sei $(\varphi, \Phi) := (\varphi, T)$ die Zeit- und $(\psi, \Psi) := (\varphi, S)$ die Platzkomplexität von Turingmaschinen. Da der Platzaufwand stets durch den Zeitaufwand beschränkt ist, gilt stets $C_{(\varphi, T)}(t) \subseteq C_{(\varphi, S)}(t)$. Nach Korollar 16.10 gibt es also zu jedem total rekursiven t ein $t' \geq t$, sodass die von Turingmaschinen mit einem Platzaufwand t' berechenbaren Funktionen genau diejenigen sind, die von Turingmaschinen mit einem Zeitaufwand t' berechnet werden können, d.h. $C_{(\varphi, T)}(t') = C_{(\varphi, S)}(t')$.

Ein weiteres zentrales Ergebnis der allgemeinen Komplexitätstheorie ist der Beschleunigungssatz (Speed-up-Theorem), der zeigt, dass es bezüglich jedes allgemeinen Komplexitätsmaßes Mengen gibt, die keine Kosten-optimalen Algorithmen besitzen.

16.13 SATZ. (SPEED-UP-THEOREM) *Sei (φ, Φ) ein allgemeines Komplexitätsmaß, und sei f eine total rekursive Funktion. Dann gibt es eine 0-1-wertige, total rekursive Funktion g mit*

$$\forall e(\varphi_e = g \Rightarrow \exists e'[\varphi_{e'} = g \ \& \ \forall x(f(\Phi_{e'}(x)) < \Phi_e(x))])$$

Wir verzichten hier auf den (recht aufwendigen) Beweis dieses Satzes.

²Man kann auf diese Forderung verzichten. Zum Beweis benötigt man dann einen simultanen Lückensatz für zwei Komplexitätsmaße.