
6. Abstrakte Komplexitätsmaße

In diesem Abschnitt führen wir den Begriff eines abstrakten Komplexitätsmaßes ein. Hierzu gehen wir axiomatisch vor: wir beobachten einige grundlegende Eigenschaften, die alle natürlichen Komplexitätsmaße haben, und legen diese der Definition eines abstrakten Komplexitätsmaßes zugrunde. Aus diesen Grundeigenschaften werden wir dann weitere Eigenschaften der abstrakten Komplexitätsmaße ableiten, womit wir diese Eigenschaften zugleich für die uns interessierenden konkreten Komplexitätsmaße erhalten.

Einem konkreten Komplexitätsmaß liegt in der Regel ein universelles Rechnermodell (in unserem Fall verschiedenen Typen von Turingmaschinen) zugrunde zusammen mit einer zugehörigen Kostenfunktion, wie Rechenzeit oder Speicherplatzbedarf. Die Universalität des Rechnermodells bedeutet, dass sich alle partiell rekursiven Funktionen (eines gegebenen Typs) berechnen lassen. Gödelisiert man die Rechner des zugrundegelegten Modells, so erhält man eine Standardaufzählung -oder wie wir auch sagen- Gödelnummerierung der partiell rekursiven Funktionen (des zugehörigen Typs) im Sinne von Definition 5.6. (Im vorhergehenden Abschnitt haben wir dies am Beispiel der normierten det. k -Band-Turingmaschinen gesehen.) In einem abstrakten Komplexitätsmaß können wir daher das zugrundeliegende Rechnermodell durch ein Gödelnummerierung φ darstellen, wobei dann der e -te Zweig φ_e der e -ten Maschine des Modells entspricht und gerade die von dieser Maschine berechnete partielle Funktion ist.

Die üblichen Kostenfunktionen haben die Eigenschaft, dass die Kosten nur für terminierende Rechnungen definiert sind, die Kosten (uniform) partiell berechenbar sind und dass man entscheiden kann, ob die Kosten einer Rechnung eine vorgegebene Höhe haben. Für Rechenzeit und Platzbedarf einer Turingmaschine M haben wir diese Eigenschaften in Lemma 3.2 und 3.7 nachgewiesen, wo wir gezeigt haben, dass $time_M(x)$ und $space_M(x)$ genau dann definiert sind, wenn $res_M(x)$ definiert ist, d.h. M bei Eingabe x terminiert, und dass die Graphen von $time_M$ und $space_M$ rekursiv sind, also Fragen der Form " $time_M(x) = n?$ " bzw. " $space_M(x) = n?$ " effektiv beantwortet werden können. Bei der Definition eines abstrakten Komplexitätsmaßes tragen wir diesen Eigenschaften der Kostenfunktionen Rechnung, indem wir der Gödelnummerierung φ eine partiell rekursive Kostenfunktion Φ mit Werten aus den natürlichen Zahlen zuordnen, deren Definitionsbereich mit dem der Gödelnummerierung übereinstimmt und deren Graph rekursiv ist. Interpretiert man dann den e -ten Zweig Φ_e von Φ als die Kostenfunktion der e -ten φ_e berechnenden Maschine, so stellt dies gerade sicher, dass diese Kostenfunktion die gewünschten Eigenschaften hat.

Da wir uns bei den konkreten Komplexitätsmaßen im Wesentlichen auf die Analyse der (1-dimensionalen) rekursiven Binärsprachen beschränken, werden wir hier entsprechend nur 0-1-wertige 1-stellige partiell rekursive Wortfunktionen über dem binären Alphabet betrachten, und die (rekursiven) Binärsprachen durch ihre (total rekursiven) charakteristischen Funktionen repräsentieren, d.h. eine Sprache $L \in \Sigma_2^*$ mit der Funktion $c_L : \Sigma_2^* \rightarrow \Sigma_2$ identifizieren, wobei $c_L(x) = 1$ falls $x \in L$ und $c_L(x) = 0$ falls $x \notin L$.

Da eine Sprache genau dann entscheidbar (rekursiv) ist, wenn ihre charakteristische Funktion berechenbar (rekursiv) ist, ist diese Identifizierung von Sprachen und 0-1-wertigen Funktion zulässig. (Wir können uns aber nicht von vornherein auf die totalen 0-1-wertigen rekursiven Funktionen, d.h. auf die Klasse der rekursiven Sprachen beschränken, da - wie wir im vorhergehenden Abschnitt gesehen haben - diese Klasse keine universellen Funktionen besitzt, in anderen Worten, universelle Rechnermodelle immer Maschinen umfassen, die nicht total sind.) In diesem Abschnitt werden wir auch verschiedentlich die Identifizierung von natürlichen Zahlen und Binärwörtern durch Interpretation des $(n+1)$ -ten Binärwortes z_n als Zahl n (und umgekehrt) vornehmen.

6.1 DEFINITION. Ein *abstraktes Komplexitätsmaß* (AKM) $\mathcal{K} = (\varphi, \Phi)$ besteht aus einer Gödelnummerierung φ der Klasse PFREK_{0-1} der 1-stelligen 0-1-wertigen partiell rekursiven Wortfunktionen über dem binären Alphabet Σ_2 zusammen mit einer partiell rekursiven Funktion $\Phi : (\Sigma_2^*)^2 \rightarrow \mathbb{N}$, wobei

$$Db(\Phi) = Db(\varphi) \quad (6.1)$$

und

$$\text{Graph}(\Phi) = \{(e, x, y) : \Phi(e, x) = y\} \text{ ist rekursiv} \quad (6.2)$$

gelten. Die Funktion Φ nennt man die *Kosten- oder Schrittzahlfunktion* des Komplexitätsmaßes \mathcal{K} und den e -ten Zweig Φ_e von Φ die *Kosten- oder Schrittzahlfunktion* des e -ten Zweiges φ_e von φ .

Ist die Kostenfunktion Φ_e an der Stelle x undefiniert, so interpretieren wir dies so, dass die Kosten zur Berechnung von $\varphi_e(x)$ unendlich hoch sind (da die Rechnung nicht terminiert), weshalb wir $\Phi_e(x) \uparrow$ mit $\Phi_e(x) = \infty$ in Grössenvergleichen identifizieren. Man beachte, dass wegen (6.2) Fragen der Form “ $\Phi_e(x) = n?$ ”, “ $\Phi_e(x) < n?$ ”, “ $\Phi_e(x) > n?$ ” u.s.w. effektiv beantwortet werden können, was wir im folgenden häufig verwenden werden.

Wie schon in der Einleitung ausgeführt, sind die von uns betrachteten konkreten Komplexitätsmaße für Turingmaschinen Beispiele von abstrakten Komplexitätsmaßen. Hierbei lassen wir jetzt auch nichtdeterministische Turingmaschinen M zur Berechnung von partiellen 0-1-wertigen Funktionen zu, wobei wir festlegen, dass $res_M(x) = 1$, falls zumindest eine Rechnung das Ergebnis 1 liefert, dass $res_M(x) = 0$, falls alle Rechnungen das Ergebnis 0 liefern, und dass $res_M(x) \uparrow$ sonst. Diese Festlegung ist so gewählt, dass die nd. Berechnung der charakteristischen Funktion c_L einer Sprache L gerade der nd. Erkennung der Sprache L entspricht, wenn wir die Ausgabe 1 mit einem Endzustand und die die Ausgabe 0 mit einem Stoppzustand, der nicht Endzustand ist, gleichsetzen.

6.2 SATZ. Sei \mathcal{M} einer der früher eingeführten Turingmaschinentypen zur Berechnung von partiellen Funktionen des Typs $\psi : \Sigma_2^* \rightarrow \Sigma_2$, sei $\ulcorner M \urcorner$ die Gödelnummer einer Maschine M vom Typ \mathcal{M} bzgl. einer Gödelisierung von \mathcal{M} und sei φ definiert durch

$$\varphi(e, x) = \begin{cases} res_M(x) & \text{falls } e = \ulcorner M \urcorner \\ \uparrow & \text{sonst.} \end{cases}$$

Dann ist φ eine Standardaufzählung von PFREK_{0-1} , und für die durch

$$\Phi_T(e, x) = \begin{cases} time_M(x) & \text{falls } e = \ulcorner M \urcorner \\ \uparrow & \text{sonst} \end{cases}$$

und

$$\Phi_S(e, x) = \begin{cases} space_M(x) & \text{falls } e = \ulcorner M \urcorner \\ \uparrow & \text{sonst} \end{cases}$$

definierten partiellen Funktionen Φ_T und Φ_S sind die Paare (φ, Φ_T) und (φ, Φ_S) abstrakte Komplexitätsmaße.

BEWEISIDEE. Aus unserer Analyse der verschiedenen Turingmaschinentypen folgt, dass die Turingmaschinen des Typs \mathcal{M} ein universelles Berechnungsmodell zur Berechnung von Funktionen des Typs $\psi : \Sigma_2^* \rightarrow \Sigma_2$ sind. Wegen der Effektivität der Gödelisierung von \mathcal{M} ist φ daher eine berechenbare schwach universelle Funktion für PFREK_{0-1} , also nach Churchscher These universell für PFREK_{0-1} . Zum Nachweis, dass φ eine Standardaufzählung von PFREK_{0-1} ist, genügt es also zu einer gegebenen partiell rekursiven Funktion $\psi : \Sigma_2^* \times \Sigma_2^* \rightarrow \Sigma_2$ eine rekursive Übersetzungsfunktion h von ψ nach φ anzugeben. Wegen der Universalität von \mathcal{M} könne wir eine Maschine M diesen Typs angeben, die die mit Hilfe der Paarfunktion kodierte Version $\psi' \in \text{PFREK}_{0-1}$ von ψ , $\psi'(1^{|e|}0ex) = \psi(e, x)$, berechnet. Da wir aus M effektiv eine Maschine M_e zur Berechnung des e -ten Zweiges $\psi_e(x) = \psi'(1^{|e|}0ex)$ von ψ angeben können (vgl. den Beweis von Korollar 5.7), folgt wegen der Effektivität der Gödelisierung und mit Churchscher These, dass die Funktion $h(e) = \ulcorner M_e \urcorner$ rekursiv ist, und nach Definition gilt gerade $\psi_e = \varphi_{h(e)}$. Dass die Funktionen Φ_T und Φ_S partiell rekursiv sind und die Eigenschaften (6.1) und (6.2) haben, folgt unmittelbar aus Lemma 3.2 und 3.7. \square

Weitere AKMs erhält man durch Betrachtung anderer universeller Berechnungsmodelle, wie z.B. Registermaschinen, und deren Zeit- und Platzbedarf. Neben Rechenzeit und Speicherbedarf kann man aber auch andere Kostenfunktionen betrachten. So erfüllt z.B. für det. 1-Band-Turingmaschinen die sog. Umkehrkomplexität, d.h. die Anzahl der Richtungswechsel des Lese-Schreib-Kopfes in einer terminierenden Rechnung die Anforderungen an eine abstrakte Kostenfunktion (Übung!). Kein AKM erhalten wir jedoch, wenn wir stets das Ergebnis einer terminierenden Rechnung als deren Kosten definieren, d.h. (φ, φ) für eine Gödelnummerierung φ betrachten, also $\Phi = \varphi$ wählen. Dann ist Φ zwar partiell rekursiv und (6.1) trivialerweise erfüllt. Aus der Unentscheidbarkeit des Halteproblems folgt jedoch, dass der Graph einer Gödelnummerierung nicht rekursiv ist, die Eigenschaft (6.2) von Kostenfunktionen hier also nicht zutrifft (Übung!). Ordnen wir einer Gödelnummerierung φ die Kostenfunktion $\Phi(e, x) = |x|$ zu, bzgl. der die Kosten einer Rechnung gerade die zugehörige Eingabengänge ist, so ist Φ wiederum partiell (sogar total) rekursiv und offensichtlich (6.2) erfüllt. Die Forderung (6.1) wird aber nicht erfüllt (da Gödelnummerierungen echt partielle Funktionen sind), sodass auch hier kein AKM vorliegt.

Neben den oben aufgeführten natürlichen Beispielen von AKMs können wir auch "pathologische" AKMs konstruieren. Zum Beispiel kann man zu jeder total rekursiven Funktion $f : \Sigma_2^* \rightarrow \Sigma_2$ ein AKM (φ', Φ') angeben, das die Berechnung von f zum "Nulltarif" erlaubt, d.h. in dem es einen Index e gibt, sodass $\varphi'_e = f$ und $\Phi'_e(x) = 0$ für alle x gilt. Hierzu geht man von einem beliebigen AKM (φ, Φ) aus, wählt einen φ -Index e für f , setzt $\varphi' = \varphi$ und definiert Φ' durch

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

Dieses Beispiel zeigt, dass Ergebnisse für einzelne AKMs i.a. nicht sehr interessant sind, da die betreffenden AKMs den üblichen Vorstellungen eines Komplexitätsmaßes nicht entsprechen müssen. Wir werden daher nur an Ergebnissen interessiert sein, die für *alle* AKMs gelten, da diese dann insbesondere für die uns interessierenden konkreten Komplexitätsmaße gelten. Interessanterweise lassen sich jedoch gewisse Ergebnisse für einzelne AKMs - mit Hilfe der unten gezeigten rekursiven Vergleichbarkeit aller AKMs - auf alle AKMs übertragen, sodass sich manchmal die Analyse geeignet gewählter pathologischer AKMs durchaus lohnen kann. Bevor wir auf dieses Phänomen näher eingehen werden, führen wir noch die zu einem AKM gehörenden (worst-case-) Komplexitätsklassen ein, die wir als *abstrakte Komplexitätsklassen* bezeichnen.

6.3 DEFINITION. Sei (φ, Φ) ein AKM und sei $s : \mathbb{N} \rightarrow \mathbb{N}$ eine total rekursive Funktion. Die (φ, Φ) -Komplexitätsklasse mit Schranke (oder Namen) $s(n)$ ist die Klasse

$$C_{(\varphi, \Phi)}(s(n)) = \{A \subseteq \Sigma_2^* : \exists e(\varphi_e = c_A \ \& \ \forall x(\Phi_e(x) \leq s(|x|)))\}.$$

Ist $S = \{s_e : e \geq 0\}$ eine uniform rekursive Familie von Schrankenfunktionen, so nennt man

$$C_{(\varphi, \Phi)}(S) = \bigcup_{e \geq 0} C_{(\varphi, \Phi)}(s_e(n))$$

eine *allgemeine (φ, Φ) -Komplexitätsklasse*.

Man beachte, dass eine abstrakte Komplexitätsklasse nur aus rekursiven Sprachen besteht, also stets $C_{(\varphi, \Phi)}(s(n))$ in REK enthalten ist. Weiter sieht man leicht ein, dass die von uns eingeführten konkreten Turingmaschinen-Komplexitätsklassen Beispiele für abstrakte Komplexitätsklassen sind.

6.4 KOROLLAR. Die Komplexitätsklassen $\text{DTIME}_{(k)}(t(n))$, $\text{NTIME}_{(k)}(t(n))$, $\text{DSPACE}_{(k)}(t(n))$ und $\text{NSPACE}_{(k)}(t(n))$ sind abstrakte Komplexitätsklassen. Entsprechend sind die Klassen PTIME, NPTIME, u.s.w. allgemeine abstrakte Komplexitätsklassen.

BEWEIS. Dies folgt unmittelbar aus Satz 6.2. Wählen wir z.B. (φ, Φ) als das abstrakte Komplexitätsmaß, das wir erhalten, wenn wir det. on-line TMs zur Berechnung 1-stelliger partieller Funktionen gödelisieren und deren Rechenzeit betrachten, so gilt gerade $C_{(\varphi, \Phi)}(t(n)) = \text{DTIME}(t(n))$. \square

Nachdem wir die Grundkonzepte der abstrakten Komplexitätstheorie eingeführt haben und gesehen haben, dass die von uns untersuchten konkreten Komplexitätsmaße und -klassen abstrakte Komplexitätsmaße und -klassen sind, wollen wir nun einige grundlegende Eigenschaften dieser Konzepte zusammenstellen. Als erstes stellen wir den schon angesprochenen Vergleichbarkeitssatz für AKMs vor.

6.5 SATZ. (VERGLEICHBARKEITSSATZ) Seien (φ, Φ) und (ψ, Ψ) AKMs und sei $h : \Sigma_2^* \rightarrow \Sigma_2^*$ eine rekursive Übersetzungsfunktion der Gödelnummerierung φ in die Gödelnummerierung ψ (d.h. $\varphi_e = \psi_{h(e)}$). Dann gibt es eine streng monotone total rekursive Funktion $g : \Sigma_2^* \times \mathbb{N} \rightarrow \mathbb{N}$, sodass

$$\forall e \ \forall x \ (\Phi_e(x) \leq g(x, \Psi_{h(e)}(x)) \ \& \ \Psi_{h(e)}(x) \leq g(x, \Phi_e(x))) \quad (6.3)$$

gilt.¹

BEWEIS. Wir definieren die Hilfsfunktion $g' : \Sigma_2^* \times \Sigma_2^* \times \mathbb{N} \rightarrow \mathbb{N}$ durch

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

und setzen

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

Um zu zeigen, dass g total und rekursiv ist, genügt es diese Eigenschaften für g' nachzuweisen. Um $g'(e, x, n) \downarrow$ zu zeigen, können wir o.B.d.A. davon ausgehen, dass $\Phi_e(x) = n$ oder $\Psi_{h(e)}(x) = n$ gilt. Da $\Phi_e = \Psi_{h(e)}$ folgt aus (6.1), dass in diesem Fall sowohl $\Phi_e(x)$ als auch $\Psi_{h(e)}(x)$ definiert sind und damit auch $g'(e, x, n)$. Dass g' berechenbar - also nach Churchscher These rekursiv - ist, sieht man wie folgt. Gegeben eine Eingabe (e, x, n) können wir wegen der Berechenbarkeit von h , Φ , und Ψ sowie der Entscheidbarkeit der Graphen der beiden letzten Funktionen (vgl. (6.2)) effektiv entscheiden, ob der erste Fall in der Definition von $g'(e, x, n)$ vorliegt, und in diesem Fall $\Phi_e(x) + \Psi_{h(e)}(x)$ berechnen.

Die strenge Monotonie von g folgt unmittelbar aus der Definition von g : Gilt $x \leq x'$ und $n \leq n'$, so folgt $g(x, n) \leq g(x', n')$ aus der Tatsache, dass die Maximumbildung im Falle von $g(x', n')$ über eine Obermenge der im Falle von $g(x, n)$ verwendeten Menge erfolgt. Dass, falls zusätzlich $x < x'$ oder $n < n'$ gilt, auch $g(x, n) < g(x', n')$ gilt, folgt aus der Tatsache, dass in der Definition von g zu dem gebildeten Maximum die Summe der beiden Eingaben addiert wird (wobei das Eingabewort x als die zugehörige Zahl interpretiert wird).

Zum Nachweis von (6.3) sei e gegeben. Es genügt dann die beiden Ungleichungen in (6.3) für Wörter x mit $e < x$ nachzuweisen, wobei wir o.B.d.A. davon ausgehen können, dass $\Phi_e(x)$ oder $\Psi_{h(e)}(x)$ definiert ist - und damit, wie oben beobachtet, in der Tat beide Terme. Es gilt dann nach Definition von g'

$$g'(e, x, \Phi_e(x)) = g'(e, x, \Psi_{h(e)}(x)) = \Phi_e(x) + \Psi_{h(e)}(x).$$

Da wegen $e < x$ nach Definition von g

$$g'(e, x, \Phi_e(x)) \leq g(x, \Phi_e(x)) \ \& \ g'(e, x, \Psi_{h(e)}(x)) \leq g(x, \Psi_{h(e)}(x))$$

folgt hieraus unmittelbar die Behauptung. \square

Als eine erste Anwendung des Vergleichbarkeitssatzes zeigen wir, dass man bezüglich eines jeden AKM zur Berechnung einer gegebenen total rekursiven Funktion ein beliebig ineffizientes Programm schreiben kann.

6.6 SATZ. (VERZÖGERUNGSSATZ) *Seien (ψ, Ψ) ein AKM, sei f eine total rekursive Funktion aus PFREK_{0-1} , und sei t eine rekursive Schrankenfunktion $t : \Sigma_2^* \rightarrow \mathbb{N}$. Dann gibt es einen ψ -Index e von f (d.h. $f = \psi_e$), sodass $\Psi_e(x) \geq t(x)$ für fast alle x gilt.*

¹Hierbei ist der Wert eines Funktionsterms undefiniert (d.h. unendlich), falls der Term einen Teilterm enthält, dessen Wert undefiniert (d.h. unendlich) ist. Diese Konvention werden wir generell verwenden, weshalb z.B. für undefiniertes $\psi(x)$ der Term $\psi(x) - \psi(x)$ ebenfalls undefiniert (und nicht etwa 0) ist.

BEWEIS. Sei (φ, Φ) das AKM, das man durch Gödelisierung der deterministischen Mehrband-Turingmaschinen und deren Rechenzeit erhält, und sei h eine Übersetzungsfunktion von φ nach ψ . Weiter sei nach dem Vergleichbarkeitssatz g eine streng monotone rekursive Funktion, für die (6.3) gilt. Wir können dann eine Turingmaschine M angeben, die f berechnet und zur Berechnung von $f(x)$ mindestens $g(x, t(x))$ Schritte benötigt. Hierzu verbraucht die Maschine M bei Eingabe x zunächst mit einer sinnlosen Aufgabe - z.B. berechnet M die Unärdarstellung von $g(x, t(x))$ auf irgendwelchen Hilfsbändern, was wegen der Rekursivität von g und t möglich ist aber bereits zum Schreiben der Ausgabe über $g(x, t(x))$ Schritte erfordert - und simuliert dann eine beliebige Turingmaschine zur Berechnung von f . Für die Gödelnummer e von M gilt dann $\varphi_e = f$ und $\Phi_e(x) \geq g(x, t(x))$ für alle Eingaben x . Da h φ nach ψ übersetzt, gilt deshalb $\psi_{h(e)} = f$ und wegen (6.3)

$$g(x, t(x)) \leq \Phi_e(x) \leq g(x, \Psi_{h(e)}(x)),$$

woraus mit der strengen Monotonie von g aber $t(x) \leq \Psi_{h(e)}(x)$ folgt. Der ψ -Index $h(e)$ hat also die gewünschten Eigenschaften. \square

Der Vergleichbarkeitssatz lässt sich auf Komplexitätsklassen wie folgt übertragen:

6.7 KOROLLAR. (VERGLEICHBARKEITSSATZ FÜR KOMPLEXITÄTSKLASSEN) *Seien (φ, Φ) und (ψ, Ψ) AKMs. Dann gibt es eine rekursive Funktion $g^* : \mathbb{N} \rightarrow \mathbb{N}$, sodass für alle rekursiven Schrankenfunktionen $s : \mathbb{N} \rightarrow \mathbb{N}$ mit $s(n) \geq n$ die (φ, Φ) -Komplexitätsklasse mit Schranke s in der (ψ, Ψ) -Komplexitätsklasse mit Schranke $g^* \circ s$ enthalten ist, d.h.*

$$C_{(\varphi, \Phi)}(s(n)) \subseteq C_{(\psi, \Psi)}(g^*(s(n))). \quad (6.4)$$

BEWEIS. Sei h eine Übersetzungsfunktion von φ nach ψ und sei g eine streng monotone rekursive Funktion, die (6.3) erfüllt. Wir behaupten, dass die durch

$$g^*(n) = \max\{g(x, n) : |x| \leq n\}$$

definierte rekursive Funktion g^* die gewünschte Eigenschaft hat. Sei also s mit $s(n) \geq n$ gegeben. Zum Nachweis von (6.4) sei A eine Sprache aus der Klasse $C_{(\varphi, \Phi)}(s(n))$. Wir müssen zeigen, dass A auch in der Klasse $C_{(\psi, \Psi)}(g^*(s(n)))$ liegt. Nach Annahme gibt es einen φ -Index e von c_A , sodass $\Phi_e(x) \leq s(|x|)$ für fast alle x gilt. Für den ψ -Index $h(e)$ von c_A gilt dann für fast alle Eingaben x

$$\Psi_{h(e)}(x) \leq g(x, \Phi_e(x)) \leq g(x, s(|x|)) \leq g^*(s(|x|)),$$

wobei die erste Ungleichung wegen (6.3), die zweite wegen der Monotonie von g und $\Phi_e(x) \leq s(|x|)$ und die dritte wegen $s(n) \geq n$ nach Definition von g^* gilt. Offensichtlich impliziert dies, dass A in $C_{(\psi, \Psi)}(g^*(s(n)))$ liegt. \square

Korollar 6.7 besagt, dass wir die Komplexitätsklassen für verschiedenen Komplexitätsmaße effektiv vergleichen können. So folgt z.B. dass es eine Funktion g gibt, sodass für alle Schranken $t(n) \geq n$, $\text{NTIME}(t(n))$ in $\text{DTIME}(g(t(n)))$ enthalten ist. Auf die noch weitgehend offenen Frage, wie groß diese Funktion g sein muss, werden wir später zurückkommen.

Als nächstes wollen wir zeigen, dass es bezüglich jeden AKM rekursive Mengen gibt, deren Komplexität beliebig hoch ist:

6.8 SATZ. Sei (φ, Φ) ein AKM und sei s eine total rekursive Schrankenfunktion $s: \mathbb{N} \rightarrow \mathbb{N}$. Dann gibt es eine total rekursive Funktion $f \in \text{PFREK}_{0-1}$, sodass für jeden φ -Index e von f gilt, dass $\Phi_e(x) > s(|x|)$ für unendlich viele Wörter x .

BEWEIS. Wir definieren eine total rekursive Funktion f mit den gewünschten Eigenschaften mit Hilfe eines Diagonalarguments. Die gesuchte Funktion muss die Anforderungen

$$\mathcal{A}_e : f = \varphi_e \Rightarrow \exists^\infty x (\Phi_e(x) > s(|x|))$$

für alle φ -Indizes e erfüllen. Die Grundüberlegung, die uns erlaubt diese Anforderungen zu erfüllen ist die folgende: Diagonalisieren wir gegen φ_e an einer Stelle x , indem wir $f(x) = 1 - \varphi_e(x)$ setzen, so sichert dies $f \neq \varphi_e$ und damit die Erfüllung von \mathcal{A}_e . Da wir gleichzeitig die Funktion f total rekursiv machen müssen, können wir jedoch nur an solchen Stellen x diagonalisieren, für die wir effektiv erkennen können, dass $\varphi_e(x)$ definiert (und damit der Wert von $\varphi_e(x)$ effektiv angebbbar) ist. Wie die Unentscheidbarkeit des Halteproblems zeigt, können wir letzteres i.a. jedoch nicht erkennen. Wir lösen dieses Dilemma, indem wir beobachten, dass wir nicht in jedem Fall die Anforderung \mathcal{A}_e durch Diagonalisierung erfüllen müssen, sondern nur im Fall, dass $\Phi_e(x) < s(|x|)$ für fast alle x gilt, also die Rechnung von $\varphi_e(x)$ für fast alle Eingaben x zu schnell ist. Solche x können wir jedoch wegen (6.2) effektiv erkennen und - wegen (6.1) - wissen wir, dass $\varphi_e(x)$ für solche x definiert ist. Wir können also gerade die Eingaben x , die der Konklusion von \mathcal{A}_e nicht genügen, zur Diagonalisierung und damit zur Zerstörung der Prämisse von \mathcal{A}_e verwenden.

Um Konflikte zwischen den einzelnen Anforderungen zu vermeiden, zerlegen wir die Menge der Binärwörter effektiv in unendlich viele unendliche Teilmengen R_e und reservieren die Wörter der e -ten Teilmenge für die Erfüllung der e -ten Anforderung. Hierzu bietet es sich an, die Eingaben von f als kodierte Paare $\langle e, x \rangle$ aufzufassen und

$$f = \varphi_e \Rightarrow \forall x (\Phi_e(\langle e, x \rangle) > s(|\langle e, x \rangle|))$$

sicherzustellen. Erweist sich für eine Eingabe $\langle e, x \rangle$ die Berechnung von $\varphi_e(\langle e, x \rangle)$ als zu schnell, d.h. gilt $\Phi_e(\langle e, x \rangle) \leq s(|\langle e, x \rangle|)$, so können wir dies, wie gerade gesehen, (wegen (6.2)) erkennen und (wegen (6.1)) dann durch die Festlegung $f(\langle e, x \rangle) = 1 - \varphi_e(\langle e, x \rangle)$ gegen φ_e an dieser Stelle effektiv diagonalisieren.

Formal ist die Funktion f wie folgt definiert.

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

Dass f die gewünschten Eigenschaften hat, ergibt sich aus unseren oben angestellten Überlegungen. \square

Dieser Satz über schwer lösbare Probleme lässt sich wie folgt verschärfen: In jedem AKM können wir zu jeder rekursiven Schranke eine Sprache angeben, deren Erkennung für *fast alle* Eingaben (nicht nur für unendlich viele Eingaben) mit Kosten oberhalb dieser Schranke verbunden ist.

6.9 SATZ. Sei (φ, Φ) ein AKM und sei s eine total rekursive Schrankenfunktion $s: \mathbb{N} \rightarrow \mathbb{N}$. Dann gibt es eine total rekursive Funktion $f \in \text{PFREK}_{0-1}$, sodass für jeden φ -Index e von f gilt, dass $\Phi_e(x) > s(|x|)$ für fast alle Wörter x .

BEWEIS. Im Vergleich zum Satz 6.8 muss die Funktion f nun die verschärften Anforderungen

$$\mathcal{A}'_e : f = \varphi_e \Rightarrow \forall x (\Phi_e(x) > s(|x|))$$

erfüllen. D.h., falls e ein φ -Index von f ist, muss Φ_e nun *fast überall* komplex sein und nicht - wie in Satz 6.8 - nur auf *unendlich vielen* Eingaben.

Grundsätzlich verfolgen wir zur Erfüllung von \mathcal{A}'_e die gleiche Strategie wie zur Erfüllung von \mathcal{A}_e im Beweis von Satz 6.8: Gilt für eine Eingabe x die Forderung $\Phi_e(x) > s(|x|)$ nicht, so können wir dies wegen (6.2) effektiv erkennen und wissen (wegen (6.1)), dass $\varphi_e(x)$ definiert ist, weshalb wir durch die Festlegung $f(x) = 1 - \varphi_e(x)$ gegen φ_e diagonalisieren und damit \mathcal{A}'_e erfüllen können.

Da wir nun schon diagonalisieren müssen, wenn $\Phi_e(x) < s(|x|)$ unendlich oft gilt, können wir den einzelnen Anforderungen jedoch nicht mehr wie im Beweis von Satz 6.8 disjunkte Eingabebereiche von f zuordnen. Dadurch kann es nun zu Konflikten zwischen den einzelnen Anforderungen kommen, nämlich dann, wenn für eine Eingabe x Indizes e und e' existieren, sodass $\Phi_e(x) < s(|x|)$ und $\Phi_{e'}(x) < s(|x|)$ gilt und $\varphi_e(x) \neq \varphi_{e'}(x)$. Wir können dann nur gegen einen der beiden Indizes an dieser Stelle diagonalisieren, da z.B. das Setzen von $f(x) = 1 - \varphi_e(x)$ impliziert, dass $f(x) = \varphi_{e'}(x)$ gilt. Wir müssen für diese Konflikte eine faire Lösung finden, die verhindert, dass im Konfliktfall immer gegen einen Index entschieden wird und damit, obwohl $\Phi_e(x) < s(|x|)$ unendlich oft gilt (und damit die Konklusion von \mathcal{A}'_e nicht erfüllt ist) keine dieser Diagonalisierungschancen ergriffen wird (die Prämisse von \mathcal{A}'_e also nicht zerstört und damit \mathcal{A}'_e insgesamt nicht erfüllt wird). Eine solche faire Lösung erhält man, indem man im Konfliktfall Anforderungen mit kleinerem Index den Vorzug gibt: Kommt nämlich eine Anforderung bei der Diagonalisierung an der Stelle x zum Zuge, so ist diese Anforderung ein für alle mal erfüllt und muss im folgenden nicht mehr beachtet werden. Da es für einen gegebenen Index e nur endlich viele kleinere Indizes gibt, kann eine Diagonalisierungschance für \mathcal{A}'_e durch diese Konfliktlösungsstrategie also nur endlich oft zu nichte gemacht werden. Da die Anforderung \mathcal{A}'_e jedoch eine Diagonalisierung nur dann erforderlich macht, wenn $\Phi_e(x) < s(|x|)$ unendlich oft gilt, also unendlich viele Diagonalisierungschancen vorliegen, verhindert die Strategie die Erfüllung der Anforderung \mathcal{A}'_e nicht.

Im folgenden beschreiben wir die Definition der Funktion f formal. Die Definition erfolgt induktiv in Stufen, wobei wir in Stufe n nicht nur den Wert von $f(z_n)$ festlegen sondern auch definieren, was es bedeutet, dass eine Anforderung \mathcal{A}'_e bei Stufe n Beachtung verlangt bzw. findet.

Beschreibung von Stufe n : Die Anforderung \mathcal{A}'_e fordert Beachtung bei Stufe n , falls $e \leq n$ (wir fassen den Index e hier als Zahl auf), \mathcal{A}'_e bei keiner der vorhergehenden Stufen Beachtung gefunden hat, und

$$\Phi_e(z_n) \leq s(|z_n|) \quad (6.5)$$

gilt. Fordert bei Stufe n eine Anforderung Beachtung, so wählen wir das kleinste e , sodass \mathcal{A}'_e Beachtung fordert, setzen

$$f(z_n) = 1 - \varphi_e(z_n) \quad (6.6)$$

und sagen, dass \mathcal{A}'_e bei Stufe n Beachtung findet. Fordert keine der Anforderungen Beachtung, so setzen wir $f(z_n) = 0$. (In diesem Fall ist es nur wichtig, dass wir $f(z_n)$ irgendwie effektiv festlegen, der gewählte Wert spielt hierbei keine Rolle.)

Um zu zeigen, dass f die gewünschten Eigenschaften hat, beobachten wir zunächst, dass f total und berechenbar, also nach Churchscher These total rekursiv ist. Die Totalität von f folgt aus (6.1) und (6.2), da wir $f(z_n)$ nur dann gemäß (6.6) festlegen, wenn (6.5) gilt, also $\Phi(z_n)$ und damit auch $\varphi(z_n)$ definiert ist. Zum Nachweis der Berechenbarkeit von f genügt es induktiv zu zeigen, dass jede Stufe n der Konstruktion effektiv ausgeführt werden kann. Um dies für Stufe n zu zeigen, gehen wir nach Induktionsvoraussetzung davon aus, dass die Stufen $0, \dots, n-1$ bereits effektiv ausgeführt und damit auch die Werte von $f(z_0), \dots, f(z_{n-1})$ berechnet wurden und bekannt ist, welche Anforderungen bei diesen Stufen Beachtung gefunden haben. Mit letzterer Information reduziert sich die Frage, ob die Anforderung \mathcal{A}'_e bei Stufe n Beachtung fordert, auf die Überprüfung, ob $e \leq n$ und (6.5) gelten. Während sich ersteres offensichtlich entscheiden lässt, ist letzteres wegen (6.2) entscheidbar. Da für eine Anforderung \mathcal{A}'_e , die Beachtung fordert, $\varphi_e(z_n)$ wegen (6.5) und (6.1) definiert ist, können wir hiermit schließlich $f(z_n)$ berechnen.

Es bleibt zu zeigen, dass jede Anforderung \mathcal{A}'_e erfüllt wird. Hierzu beobachten wir, dass eine Anforderung \mathcal{A}'_e höchstens einmal Beachtung findet, da eine Anforderung, sobald sie Beachtung gefunden hat, keine Beachtung mehr fordert. Weiter ist eine Anforderung \mathcal{A}'_e , die bei einer Stufe n Beachtung findet erfüllt, da nach Konstruktion $\varphi_e(z_n) \neq f(z_n)$ gilt.

Gehen wir also von der Widerspruchsannahme aus, dass die Anforderung \mathcal{A}'_e nicht erfüllt ist. Dann gilt

$$f = \varphi_e \ \& \ \exists^\infty n (\Phi_e(z_n) \leq s(|z_n|)), \quad (6.7)$$

und, wie wir gerade beobachtet haben, findet \mathcal{A}'_e nie Beachtung. Es folgt, dass \mathcal{A}'_e bei unendlich vielen Stufen Beachtung fordert, nämlich bei allen Stufen n mit $e \leq n$ und $\Phi_e(z_n) \leq s(|z_n|)$. Da \mathcal{A}'_e aber nie Beachtung findet, muss nach Definition bei jeder dieser Stufen eine Anforderung $\mathcal{A}'_{e'}$ mit $e' < e$ Beachtung finden. Da es nur endlich viele e' mit $e' < e$ gibt und da jede Anforderung höchstens einmal Beachtung findet, ist dies jedoch nicht möglich.

Die Konstruktion einer Funktion oder Menge in Stufen (d.h. induktive Definition), bei der eine unendliche Liste von Anforderungen durch Diagonalisierung zu erfüllen ist und bei der man mögliche Konflikte zwischen Anforderungen dadurch löst, dass man Anforderungen, die früher in der Liste stehen (also kleineren Index haben), bevorzugt nennt man ein *Prioritätsargument*. Die Prioritätsmethode ist eine grundlegende Diagonalisierungstechnik in der Berechenbarkeitstheorie. \square

Durch eine Verfeinerung des vorhergehenden Beweises können wir ein weiteres Beispiel eines allgemein anzutreffenden Phänomens vorstellen, nämlich das *Speed-up-Theorem*, das besagt, dass es Sprachen gibt, die keine optimalen Entscheidungsverfahren besitzen. In der Tat kann man zu jeder streng monotonen rekursiven Funktion g -egal wie schnell diese wächst; z.B. $g(n) = 2^n$ oder $g(n) = 2^{2^n}$, usw.- eine Sprache L angeben, sodass es zu jeder Maschine M , die L entscheidet, eine Maschine M' gibt, die L ebenfalls entscheidet, und für fast alle Eingaben die Kosten der ursprünglichen Maschine M um den Faktor g höher liegen als die Kosten der neuen Maschine M' (d.h. $\Phi_{M'} \succ g(\Phi_M)$ f.ü.).

6.10 SATZ. (BESCHLEUNIGUNGSSATZ) Sei (φ, Φ) ein AKM und seien $g : \mathbb{N} \rightarrow \mathbb{N}$ und $h : \mathbb{N} \rightarrow \mathbb{N}$ total rekursive Funktionen, wobei g streng monoton sei. Dann gibt es eine total rekursive Funktion $f \in \text{PFREK}_{0-1}$, sodass

$$\forall e [\varphi_e = f \Rightarrow \forall x (\Phi_e(x) \geq h(|x|))]. \quad (6.8)$$

und

$$\forall e[\varphi_e = f \Rightarrow \exists e'(\varphi_{e'} = f \ \& \ \forall x(g(\Phi_{e'}(x)) < \Phi_e(x)))] \quad (6.9)$$

gelten.

Der Beschleunigungssatz ist - wie die meisten hier vorgestellten Ergebnisse über AKMs - eher theoretischer Natur. Das Speed-Up-Phänomen trifft auf natürliche Sprachen i.a. nicht zu. Hierzu beachte man auch, dass Sprachen, die einen hohen Speed-up zulassen, selbst sehr komplex sein müssen. Erlaubt z.B. eine nichttriviale Sprache L eine einfach exponentielle Beschleunigung, so können die Kosten einer Maschine M , die L entscheidet, nicht durch eine endliche Iteration der Exponentialfunktion beschränkt werden (s. auch die Übungen hierzu).

BEWEISSKIZZE. Wir skizzieren den Beweis. Zunächst beobachten wir, dass es wegen des Vergleichbarkeitssatzes genügt, den Satz für ein spezielles AKM zu beweisen (s. Übungen). Wir tun dies hier für die Platzkomplexität deterministischer off-line 1-Band-Turingmaschinen und gehen im folgenden davon aus, dass (φ, Φ) das zugehörige AKM sei. Weiter beobachten wir, dass eine Vergrößerung des Beschleunigungsfaktors g zu einer Verschärfung des Satzes führt, weshalb wir o.B.d.A. davon ausgehen dürfen, dass $g(n) > h(n)$ und $g(n) > 2^n$ stets gilt.

Um die gewünschte Funktion f zu definieren, definieren wir zunächst eine uniform rekursive Familie $\{s_e(n) : e \geq 0\}$ von geeigneten Schrankenfunktionen, für die

$$\forall e \forall n (s_e(n) = g(s_{e+1}(n))) \quad (6.10)$$

gilt, d.h. die die Eigenschaft haben, dass die Schranken mit wachsendem Index e sukzessive um den Faktor g kleiner werden, und konstruieren f dann so, dass es die Bedingungen

$$\forall e (f = \varphi_e \Rightarrow \forall x (\Phi_e(x) > s_e(|x|))) \quad (6.11)$$

und

$$\forall e \exists e' (f = \varphi_{e'} \ \& \ \forall x (\Phi_{e'}(x) \leq s_e(|x|))) \quad (6.12)$$

erfüllt.

Dies genügt, um (6.9) zu erfüllen: Gilt $\varphi_e = f$, so gilt (wegen (6.11)) $s_e(|x|) < \Phi_e(x)$ fast überall, während es (wegen (6.12) angewandt auf $e+1$) einen φ -Index e' von f mit $\Phi_{e'}(x) \leq s_{e+1}(|x|)$ gibt, woraus (wegen (6.10) und der strengen Monotonie von g)

$$g(\Phi_{e'}(x)) \leq g(s_{e+1}(|x|)) = s_e(|x|) < g(\Phi_e(x))$$

für fast alle x folgt. Da wir die Funktionen s_e so wählen werden, dass

$$\forall e \forall n (g(n) \leq s_e(n)) \quad (6.13)$$

gilt, stellt (6.11) zusätzlich sicher, dass wegen $h(n) < g(n)$ auch (6.8) erfüllt ist.

Die Schranken $s_e(n)$ definieren wir durch geeignete Iteration der Funktion g . Sei die k -te Iteration $g^k(n)$ von $g(n)$ induktiv durch $g^0(n) = n$ und $g^{k+1}(n) = g(g^k(n))$ gegeben. Dann definieren wir $s_e(n)$ durch

$$s_e(n) = g^{n-e}(n), \quad (6.14)$$

wobei wir negative Werte von $n-e$ als 0 interpretieren. Offensichtlich impliziert dies (6.10), da $g^{n-e}(n) = g(g^{n-(e+1)}(n))$ für $e < n$.

Um nun eine rekursive Funktion f zu definieren, die die Bedingung (6.11) erfüllt, variieren wir den Beweis von Satz 6.9. Es genügt die dort betrachteten Anforderungen \mathcal{A}'_e durch die Anforderungen

$$\mathcal{A}_e^* : f = \varphi_e \Rightarrow \forall x (\Phi_e(x) > s_e(|x|))$$

zu ersetzen. Die n -te Stufe der Konstruktion sieht also nun wie folgt aus:

Beschreibung von Stufe n : Die Anforderung \mathcal{A}_e^* fordert Beachtung bei Stufe n , falls $e \leq n$, \mathcal{A}_e^* bei keiner der vorhergehenden Stufen Beachtung gefunden hat, und

$$\Phi_e(z_n) \leq s_e(|z_n|) \quad (6.15)$$

gilt. Fordert bei Stufe n eine Anforderung Beachtung, so wählen wir das kleinste e , sodass \mathcal{A}_e^* Beachtung fordert, setzen

$$f(z_n) = 1 - \varphi_e(z_n) \quad (6.16)$$

und sagen, dass \mathcal{A}_e^* bei Stufe n Beachtung findet. Fordert keine der Anforderungen Beachtung, so setzen wir $f(z_n) = 0$.

Wie im Beweis von Satz 6.9 zeigt man dann, dass f rekursiv ist und alle Anforderungen erfüllt sind, weshalb (6.11) gilt.

Es bleibt zu zeigen, dass für die so definierte Funktion f die Bedingung (6.12) ebenfalls erfüllt ist. Hierbei benutzen wir, dass wir die Platzkomplexität deterministischer 1-Band-TMs als Kostenmaß (φ, Φ) zugrundegelegt haben. Weiter setzen wir voraus, dass die Funktion g neben der bereits gemachten Annahme $g(n) > 2^n$ noch die Eigenschaft hat, dass die Berechnung von $g(n)$ nicht mehr Platz als $g(n)$ erfordert. Wir können dies o.B.d.A. annehmen, da man zu jeder rekursiven Funktion g eine grössere rekursive Funktion g^* mit dieser Eigenschaft finden kann. (Hierzu betrachtet man eine 1-Band-Turingmaschine M , die bei Eingabe x die Unärdarstellung von $g(|x|)$ berechnet und auf eine Spur ihres Bandes schreibt, dabei alle während der Rechnung benutzten Felder markiert, und anschließend (ohne zusätzliche Felder zu benutzen) die Anzahl $g^*(|x|)$ der benutzten Felder ausgibt. Dabei hängt die Rechnung von M nur von der Eingabelänge ab. Offensichtlich hat dann $g^*(n)$ die gewünschten Eigenschaften.)

Nach diesen Vorüberlegungen analysieren wir den Platzbedarf einer Turingmaschine M zur Berechnung von f , die wir durch geeignete Formalisierung der Konstruktion von f erhalten. Wegen des Bandreduktionssatzes spielt dabei die Anzahl der Bänder der Maschine keine Rolle, und wegen der Annahme $g(n) > 2^n$ können wir den Platzbedarf einfacher Teilrechnungen vernachlässigen.

Bei Eingabe $x = z_n$ berechnet M aus x die Zahl n (Platzbedarf vernachlässigbar) und simuliert dann die ersten $n + 1$ Stufen der Konstruktion. Zur Ausführung der Stufe n muss die Maschine sich aus den vorhergehenden Stufen lediglich merken, welche Anforderungen dort Beachtung fanden. (Der Platz zum Speichern dieser Information ist wiederum vernachlässigbar.) Da der Platz, der bei einer Stufe verwendet wird, bei der nächsten Stufe wiederverwendet werden kann, genügt es den Platzbedarf der Stufe n für sich zu analysieren. Hier müssen wir zunächst bestimmen, welche Anforderungen \mathcal{A}_e^* Beachtung fordern, wobei es wiederum genügt, den Platzbedarf für festes e zu bestimmen. Den Aufwand zur Beantwortung der Frage, ob $|e| \leq n$ gilt und ob \mathcal{A}_e^* bei einer früheren Stufe Beachtung fand, können wir vernachlässigen. Die Komplexität dieser Frage wird als durch den Aufwand zur Beantwortung von (6.15) bestimmt. Dieser

Platzbedarf ist durch $s_e(|z_n|)$ beschränkt: Wie unsere Vorüberlegungen gezeigt haben, können wir $s_e(|z_n|)$ innerhalb der Platzschränke $s_e(|z_n|)$ berechnen. Ohne zusätzlichen Platzbedarf können wir dann das Intervall $[-s_e(|z_n|), s_e(|z_n|)]$ auf dem Arbeitsband der e -ten Maschine markieren und prüfen, ob die Berechnung von $\varphi_e(z_n)$ mit diesem Platz auskommt, also (6.15) gilt. Da nur Anforderungen \mathcal{A}_e^* mit $e \leq n$ Beachtung fordern können, kann der Platzbedarf zur Bestimmung des Index e der Anforderung, die bei Stufe n Beachtung findet (bzw. zur Feststellung, dass solch ein e nicht existiert) also nach oben durch

$$\max\{s_0(|z_n|), \dots, s_n(|z_n|)\} \quad (6.17)$$

abgeschätzt werden. Mit dieser Information kann man dann (wegen (6.15)) den Wert von $f(z_n)$ gemäß (6.16) mit Platzschränke $s_e(|z_n|)$ berechnen, wobei e der Index der Anforderung ist, die Beachtung findet. Da auch hier früher verwendeter Platz wiederverwendet werden kann, können wir die Platzkomplexität zur Berechnung von $f(z_n)$ insgesamt durch (6.17) beschränken. Wegen (6.10) fällt diese Schranke jedoch gerade mit $s_0(|z_n|)$ zusammen, die Maschine M ist also $s_0(n)$ -platzbeschränkt.

Durch Betrachtung geeigneter Varianten dieser Turingmaschine M können wir (6.12) nun beweisen. Zu gegebenem e müssen wir eine Variante M_e von M angeben, deren Platzbedarf f.ü. durch $s_e(n)$ beschränkt ist: Für die Gödelnummer e' von M_e gilt dann $\Phi_{e'}(x) \leq s_e(|x|)$ f.ü. Wir erhalten M_e dadurch, dass wir die kleinste Stufe n_e festhalten, nach der keine Anforderung $\mathcal{A}_{e'}^*$ mit $e' < e$ mehr Beachtung findet. (Da jede Anforderung höchstens einmal Beachtung findet und es nur endlich viele e' mit $e' < e$ gibt, muss diese Stufe existieren!) Wir versorgen M_e mit der Information: n_e , den Werten von f für die endlich vielen Eingaben z_n mit $n \leq n_e$ und der Liste der Anforderungen, die bis einschließlich Stufe n_e Beachtung finden. (Diese endliche Information wird in die Zustände von M_e kodiert.) Für Eingaben z_n mit $n \leq n_e$ benutzt dann M_e diese Information, um $f(z_n)$ mit vernachlässigbarem Platzbedarf zu berechnen. Für Eingaben z_n mit $n > n_e$ simuliert M_e die Maschine M ab Stufe $n_e + 1$ bis Stufe n (unter Verwendung der in den Zuständen gespeicherten relevanten Information, welche Anforderungen bei den Stufen $\leq n_e$ Beachtung fanden), ignoriert aber bei der Überprüfung, welche Anforderungen Beachtung fordern, die ersten e Anforderungen $\mathcal{A}_{e'}^*$ mit $e' < e$. Da nach Wahl von n_e diese Anforderungen nach Stufe n_e keine Beachtung mehr fordern (da andernfalls, wegen der Minimalität, eine dieser Anforderungen auch Beachtung finden würde, was nach Wahl von n_e jedoch nicht der Fall ist), ist diese Überprüfung nämlich überflüssig. Hierdurch entfallen bei der Platzbedarfbestimmung in (6.17) die ersten e Einträge $s_0(|z_n|), \dots, s_{e-1}(|z_n|)$ und wir erhalten daher mit (6.15) die obere Platzschränke

$$\max\{s_e(|z_n|), \dots, s_n(|z_n|)\} = s_e(|z_n|)$$

zur Ausführung von Stufe n und damit die gewünschte Platzschränke $s_e(n)$ für M_e . \square

Die Existenz schwer lösbarer Probleme in allen AKMs zeigt, dass sich jede Komplexitätsklasse $C_{(\varphi, \Phi)}(s(n))$ durch Vergrößerung der Schrankenfunktion zu einer echt grösseren Komplexitätsklasse $C_{(\varphi, \Phi)}(s'(n))$ erweitern lässt: Hierzu wählt man nach Satz 6.8 eine totale Funktion $f \in \text{PFREK}_{0-1}$, die nicht in $C_{(\varphi, \Phi)}(s(n))$ liegt, zusammen mit einem φ -Index e von f und setzt $s'(n) = s(n) + \max\{\Phi_e(x) : |x| = n\}$. Der folgende *Lückensatz* zeigt jedoch, dass es keine rekursive Funktion g gibt, sodass die Vergrößerung einer jeden Schranke s um den Faktor g (d.h. der Übergang von s zu $g \circ s$) zu einer echt grösseren Komplexitätsklasse führt (selbst wenn wir hierbei nur Schranken s betrachten, die oberhalb einer vorgegebenen Schranke liegen). Der Grund

hierfür ist, dass die Kostenfunktionen nicht dicht in den partiell rekursiven Funktionen liegen, sondern es für jedes rekursives g Folgen von Intervallen der Länge g gibt, die keine Kostenfunktionen enthalten.

6.11 SATZ. (LÜCKENSATZ) Sei $\mathcal{K} = (\varphi, \Phi)$ ein AKM und seien $g : \mathbb{N} \rightarrow \mathbb{N}$ und $h : \mathbb{N} \rightarrow \mathbb{N}$ total rekursive Funktionen, wobei $g(n) \geq n$ für alle Zahlen n gelte. Dann gibt es eine total rekursive Funktion $f : \mathbb{N} \rightarrow \mathbb{N}$ mit $f \geq h$ und

$$\forall e [\forall x (\Phi_e(x) \leq g(f(|x|))) \Rightarrow \forall x (\Phi_e(x) \leq f(|x|))] \quad (6.18)$$

D.h. keine Funktion $k : \Sigma_2^* \rightarrow \mathbb{N}$ mit Werten in der Intervallfolge $(f(|x|), g(f(|x|)))$ (d.h. $f(|x|) \leq k(|x|) \leq g(f(|x|))$) für fast alle Binärwörter x ist eine Schrittzahlfunktion Φ_e bzgl. \mathcal{K} , weshalb insbesondere

$$C_{(\varphi, \Phi)}(f(n)) = C_{(\varphi, \Phi)}(g(f(n))) \quad (6.19)$$

gilt.

BEWEIS. Wir definieren die gesuchte Funktion f induktiv. Um (6.18) zu erfüllen, stellen wir sicher, dass für jede Zahl n keine der ersten $n+1$ Schrittzahlfunktionen Φ_e für Eingaben dieser Länge Werte in dem Intervall $(f(n), g(f(n)))$ annimmt, d.h.

$$\forall e \leq n \forall x [|x| = n \Rightarrow (\Phi_e(x) \leq f(n) \vee g(f(n)) < \Phi_e(x))] \quad (6.20)$$

für alle $n \in \mathbb{N}$ gilt (wobei wir hier und im folgenden den Index e als natürliche Zahl auffassen, indem wir das Binärwort z_m mit der Zahl m identifizieren). Dies sichert, dass die e -te Kostenfunktion Φ_e für allen Eingaben x deren Länge mindestens e ist, das Intervall $(f(|x|), g(f(|x|)))$ meidet, was offensichtlich (6.18) impliziert.

Um (6.20) zu erfüllen, beobachten wir, dass es für jede Zahl n nur endlich viele Werte $\Phi_e(x)$ mit $e \leq n$ und $|x| = n$ gibt, da es nur $n+1$ natürliche Zahlen $\leq n$ und 2^n Binärwörter der Länge n gibt. Die Menge

$$D_n = \{\Phi_e(x) : e \leq n \ \& \ |x| = n \ \& \ \Phi_e(x) \downarrow\}$$

ist also endlich, und wenn wir $d(n)$ als das Maximum von D_n wählen und $f(n)$ wiederum als das Maximum von $d(n)$ und $h(n)$ wählen (um der Forderung $f(n) \geq h(n)$ zu genügen), so ist (6.20) offensichtlich erfüllt. (Für die betroffenen e und x gilt nämlich für $\Phi_e(x) \downarrow$, dass $\Phi_e(x) \leq f(n)$, und für $\Phi_e(x) \uparrow$ interpretieren wir dies als $\Phi_e(x) = \infty$, weshalb in diesem Fall $g(f(n)) < \Phi_e(x)$ gilt.)

Die derart definierte Funktion f ist jedoch nicht rekursiv, da wir nicht entscheiden können, welche $\Phi_e(x)$ definiert sind, und daher das Maximum $d(n)$ von D_n nicht berechnen können. Die angestellten Überlegungen zeigen jedoch, dass es für jede Zahl n Werte m mit $m \geq h(n)$ gibt, für die

$$\forall e \leq n \forall x [|x| = n \Rightarrow (\Phi_e(x) \leq m \vee g(m) < \Phi_e(x))] \quad (6.21)$$

gilt, und zur Erfüllung von (6.20) genügt es einen dieser Werte m als Funktionswert $f(n)$ zu wählen.

Wegen der Rekursivität des Graphen der Kostenfunktion Φ , können wir jedoch für ein gegebenes m entscheiden, ob es (6.21) erfüllt, da wir hierzu nur für endlich viele e und x entscheiden müssen, ob für das gegebene m und das hieraus -wegen der Rekursivität von g - berechenbare $g(m)$ eine der beiden Beziehungen $\Phi_e(x) \leq m$ oder

$g(m) < \Phi_e(x)$ gilt. Folglich können wir auch das kleinste m oberhalb von $h(n)$ mit dieser Eigenschaft finden, indem wir sukzessive für $m = h(n)$, $m = h(n) + 1$, ... testen, ob (6.21) gilt. Die durch

$$f(n) = \min\{m : m \geq h(n) \ \&\forall e \leq n \ \forall x[|x| = n \Rightarrow (\Phi_e(x) \leq m \vee g(m) < \Phi_e(x))]\}$$

definierte Funktion f ist daher rekursiv und hat die gewünschten Eigenschaften. \square

Wenden wir den Lückensatz auf die Zeitkomplexität deterministischer Turingmaschinen an und wählen $h(n) = n^2$ und $g(n) = 2^n$, so sehen wir, dass es hyperlineare rekursive Zeitschranken $t(n)$ gibt, sodass

$$\text{DTIME}(t(n)) = \text{DTIME}(2^{t(n)})$$

gilt. Wie wir im nächsten Abschnitt sehen werden, kann solch eine große Lücke dort jedoch nicht für natürliche Schranken $t(n)$ auftreten, weshalb der Lückensatz weniger von praktischer Bedeutung als von theoretischem Interesse ist, indem er ein prinzipiell bestehendes Phänomen aufzeigt, dem man bei praktischen Anwendungen jedoch nicht begegnet. Beim Beweis allgemeiner Sätze über ein Komplexitätsmaß muss man diese Phänomene jedoch berücksichtigen und in der Regel durch geeignete Zusatzannahmen eliminieren.

Mit Hilfe des Vergleichbarkeitssatzes können wir eine weiter interessante Folgerung aus dem Lückensatz ziehen.

6.12 KOROLLAR. *Seien (φ, Φ) und (ψ, Ψ) AKMs, sodass für jede rekursive Schranke $s : \mathbb{N} \rightarrow \mathbb{N}$*

$$C_{(\varphi, \Phi)}(s(n)) \subseteq C_{(\psi, \Psi)}(s(n)) \quad (6.22)$$

*gilt.*² *Dann gibt es zu jeder rekursive Funktion $h : \mathbb{N} \rightarrow \mathbb{N}$ eine rekursive Schranke $s' : \mathbb{N} \rightarrow \mathbb{N}$ mit $h \leq s'$, sodass*

$$C_{(\varphi, \Phi)}(s'(n)) = C_{(\psi, \Psi)}(s'(n)) \quad (6.23)$$

gilt.

BEWEIS. Nach dem Vergleichbarkeitssatz für Komplexitätsklassen (Korollar 6.7 mit vertauschten Rollen von (φ, Φ) und (ψ, Ψ)) gibt es eine rekursive Funktion g^* , sodass für alle rekursiven Schranken $s(n)$ oberhalb von h (wobei wir o.B.d.A. davon ausgehen, dass $h(n) \geq n$)

$$C_{(\psi, \Psi)}(s(n)) \subseteq C_{(\varphi, \Phi)}(g^*(s(n))) \quad (6.24)$$

gilt. Andererseits liefert eine Anwendung des Lückensatzes auf das Maß (ψ, Ψ) und die Funktion $g = g^*$ eine Funktion $f \geq h$ mit

$$C_{(\psi, \Psi)}(f(n)) = C_{(\psi, \Psi)}(g^*(f(n))).$$

Setzen wir dieses f in (6.24) ein, erhalten wir also

$$C_{(\psi, \Psi)}(g^*(f(n))) \subseteq C_{(\varphi, \Phi)}(g^*(f(n))).$$

²Diese Zusatzannahme ist nicht notwendig. Sie lässt sich mit Hilfe eines simultanen Lückensatzes für zwei AKMs (φ, Φ) und (ψ, Ψ) eliminieren.

Da die Umkehrung nach Annahme (6.22) allgemein gilt, folgt hieraus, dass (6.23) für $s'(n) = g^*(f(n))$ erfüllt ist. \square

Da jede deterministische Zeitklasse in der korrespondierenden nichtdeterministischen Zeitklasse enthalten ist, impliziert Korollar 6.12, dass es zu jeder rekursiven Zeitschranke $t(n)$ eine Schranke $t'(n)$ mit $t \leq t'$ gibt, sodass

$$\text{DTIME}(t'(n)) = \text{NTIME}(t'(n))$$

gilt. Ebenso sieht man, dass man zu jeder rekursiven Zeitschranke $t(n)$ eine Schranke $t'(n)$ mit $t \leq t'$ erhält, für die

$$\text{DTIME}_1(t'(n)) = \text{DTIME}(t'(n))$$

gilt, der quadratische Zeitverlust bei der Simulation einer Mehrband-TM durch eine 1-Band-TM, den wir für lineare Zeitschranken nachgewiesen haben, also nicht bei jeder Zeitschranke auftritt. Die Schranken $t'(n)$, die wir hier erhalten sind jedoch alles andere als natürlich, und die Frage, ob diese Gleichheiten auch für interessante, für die Praxis relevante Schranken zu erzielen sind, ist ein offenes Problem.

Wir beschließen unseren Exkurs in die abstrakte Komplexitätstheorie mit der Beobachtung, dass sich allgemeine Komplexitätsklassen als einfache Komplexitätsklassen beschreiben lassen, falls die Folge der Schrankenfunktionen streng monoton ist.

6.13 SATZ. (VEREINIGUNGSSATZ) Sei (φ, Φ) ein AKM und sei $S = \{s_e : e \geq 0\}$ eine uniform rekursive Familie von Schrankenfunktionen, wobei

$$s_e(n) < s_{e+1}(n) \tag{6.25}$$

für alle $e, n \in \mathbb{N}$ gelte. Dann gibt es eine rekursive Schrankenfunktion $s : \mathbb{N} \rightarrow \mathbb{N}$ mit

$$C_{(\varphi, \Phi)}(s(n)) = C_{(\varphi, \Phi)}(S) \tag{6.26}$$

BEWEIS. Bevor wir die Funktion s formal definieren, beschreiben wir zunächst die Ideen, die der Definition zugrundeliegen.

Definieren wir die *Diagonale* $\hat{s} : \mathbb{N} \rightarrow \mathbb{N}$ von S durch

$$\hat{s}(n) = s_n(n),$$

so ist \hat{s} rekursiv und für jedes e gilt (wegen (6.25)) $\hat{s}(n) > s_e(n)$ für fast alle n , nämlich für alle $n > e$. Es gilt daher

$$C_{(\varphi, \Phi)}(s_e(n)) \subseteq C_{(\varphi, \Phi)}(\hat{s}(n))$$

für alle e und damit

$$C_{(\varphi, \Phi)}(S) \subseteq C_{(\varphi, \Phi)}(\hat{s}(n)).$$

Wir können jedoch nicht davon ausgehen, dass auch die Umkehrung hiervon gilt, da es wegen (6.25) Funktionen φ_e geben mag, deren Kostenfunktion Φ_e zwar unterhalb der Diagonalen \hat{s} bleiben aber jede der Einzelschranken s_e schließlich überschreiten.

Wir können dies jedoch vermeiden, wenn wir der Diagonalen erlauben, langsamer anzuwachsen, d.h. unendlich oft zurückzufallen, also an unendlich vielen Stelle n statt des Wertes $s_n(n)$ den Wert $s_m(n)$ für ein m , das kleiner als n ist, anzunehmen. Dies

werden wir bei der Definition von s ausnutzen. D.h. wir werden eine rekursive Funktion $l : \mathbb{N} \rightarrow \mathbb{N}$ mit $l(n) \leq n$ definieren, sodass die Funktion

$$s(n) = s_{l(n)}(n) \quad (6.27)$$

die gewünschten Eigenschaften hat. Da wegen der uniformen Rekursivität von S die Rekursivität von s aus der von l folgt, genügt es hierzu zu zeigen, dass

$$C_{(\varphi, \Phi)}(s_e(n)) \subseteq C_{(\varphi, \Phi)}(s(n)) \quad (6.28)$$

für alle e und

$$C_{(\varphi, \Phi)}(s(n)) \subseteq \bigcup_{e \geq 0} C_{(\varphi, \Phi)}(s_e(n)) \quad (6.29)$$

gilt.

Um ersteres sicherzustellen, darf s auf jede der Schranken s_e nur endlich oft zurückfallen, d.h. die Funktion l muss

$$\liminf_{n \rightarrow \infty} l(n) = \infty \quad (6.30)$$

erfüllen. Dies impliziert nach (6.25), dass für jedes e

$$s_e(n) \leq s_{l(n)}(n) = s(n)$$

für fast alle n gilt und damit (6.28).

Um (6.29) zu erfüllen, versuchen wir jedem Zweig φ_e der Gödelnummerierung φ eine Schrankenfunktion $s_{\sigma(e)}$ zuzuordnen, wobei wir mit der Schranke $s_{\sigma(e)} = s_e$ beginnen, und dann induktiv für alle Eingaben x prüfen, ob $\Phi_e(x)$ dieser Schranke genügt, d.h. $\Phi_e(x) \leq s_{\sigma(e)}(|x|)$ gilt. Sehen wir, dass Φ_e die Schranke überschreitet, so setzen wir $\sigma(e)$ hoch und versuchen es mit der vergrößerten Schranke. Ist φ_e die charakteristische Funktion einer Sprache A , die nicht in $C_{(\varphi, \Phi)}(S)$ liegt, so überschreitet die Kostenfunktion Φ_e jede der Schranken s_m unendlich oft, weshalb wir die Schranke $s_{\sigma(e)}$ unendlich oft hochsetzen müssen. Lassen wir nun jedesmal, wenn wir $\sigma(e)$ hochsetzen, l an dieser Stelle auf den alten Wert von $\sigma(e)$ zurückfallen, so garantiert dies, dass Φ_e auch die Schranke s unendlich oft überschreitet und daher A auch nicht in $C_{(\varphi, \Phi)}(s(n))$ liegt. Die Tatsache, dass wir e als Startwert für $\sigma(e)$ gewählt haben, stellt sicher, dass dieses Vorgehen mit (6.30) kompatibel ist.

Zur formalen Definition von l und damit von s definieren wir zunächst $\sigma(e, n)$, wobei dies den Wert von $\sigma(e)$ bezeichnen wird, den wir nach Betrachtung aller Wörter x der Länge $< n$ gewählt haben. Die Definition ist induktiv über n : Für $n \leq e$ setzen wir

$$\sigma(e, 0) = \sigma(e, 1) = \dots = \sigma(e, e) = e$$

und für $n \geq e$

$$\sigma(e, n+1) = \begin{cases} \sigma(e, n) + 1 & \text{falls es ein Wort } x \text{ der Länge } n \text{ gibt mit } \Phi_e(x) > s_{\sigma(e, n)}(n) \\ \sigma(e, n) & \text{sonst} \end{cases}$$

Man beachte, dass wegen der uniformen Rekursivität von S und der Rekursivität des Graphen von Φ die Funktion $\sigma : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ total rekursiv ist, dass $\sigma(e, n) \leq n$ für all n mit $n \geq e$ gilt, und dass $\sigma(e, n)$ schwach monoton in n ist.

Die Funktion l wird dann mit Hilfe von σ wie folgt induktiv definiert:

$$l(0) = 0$$

$$l(n+1) = \begin{cases} \min\{\sigma(e,n) : e \leq n \text{ \& } \sigma(e,n) < \sigma(e,n+1)\} & \text{falls es ein } e \leq n \text{ mit} \\ & \sigma(e,n) < \sigma(e,n+1) \text{ gibt} \\ n+1 & \text{sonst} \end{cases}$$

Man beachte wiederum, dass l rekursiv ist und $l(n) \leq n$ für alle n gilt.

Schließlich definieren wir die Funktion s mit Hilfe von l gemäß (6.27). Die Rekursivität von l überträgt sich dann auf s . Zu zeigen bleibt daher, dass s die Bedingungen (6.30) (und damit (6.28)) sowie (6.29) erfüllt.

Zum Nachweis von (6.30) sei m gegeben. Wir müssen zeigen, dass es nur endlich viele n mit $l(n+1) \leq m$ gibt. Hierzu beobachten wir, dass für ein n mit $n > m$ die Ungleichung $l(n+1) \leq m$ nach Definition von l nur dann gelten kann, wenn es ein e gibt mit $\sigma(e,n) < \sigma(e,n+1)$ und $\sigma(e,n) \leq m$. Da $\sigma(e,n)$ schwach monoton in n ist und $\sigma(e,0) = e$ gilt, muss dann $e \leq m$ gelten und für jedes solches e kann es höchstens $m+1$ Stufen n geben, bei denen $\sigma(e,n) < \sigma(e,n+1)$ und $\sigma(e,n) \leq m$ gilt. Hieraus folgt aber, dass $l(n) > m$ für fast alle n gelten muss.

Zum Nachweis von (6.29) genügt es zu zeigen, dass jede Kostenfunktion Φ_e , die von s fast überall beschränkt wird, auch von einer der Funktionen s_m beschränkt wird. Wir zeigen dies durch Kontraposition: Sei e gegeben, sodass es für jedes m unendlich viele Wörter x mit $\Phi_e(x) > s_m(|x|)$ gibt. Wir haben zu zeigen, dass $\Phi_e(x) > s(|x|)$ dann ebenfalls unendlich oft gilt. Hierzu betrachten wir die Funktion $\sigma(e,n)$. Es genügt zu zeigen dass, $\sigma(e,n) < \sigma(e,n+1)$ für unendlich viele n gilt und dass es für jedes solches n ein Wort x der Länge n mit $\Phi_e(x) > s(|x|)$ gibt.

Um ersteres zu zeigen, gehen wir von der Widerspruchsannahme aus, dass $\sigma(e,n)$ beschränkt ist. Dann gilt wegen der schwachen Monotonie von $\sigma(e,n)$ in n , dass - für geeignetes m - $\sigma(e,n) = m$ für alle hinreichend großen n gilt. Da nach Annahme $\Phi_e(x) > s_m(|x|)$ für unendlich viele x gilt, muss es aber dann ein derartiges Wort x geben, dessen Länge $n = |x|$ hinreichend groß ist, sodass $\sigma(e,n) = m$. Nach Definition von σ impliziert dies jedoch $\sigma(e,n+1) = \sigma(e,n) + 1 = m + 1$ im Widerspruch zur Wahl von m .

Es bleibt zu zeigen, dass $\sigma(e,n) < \sigma(e,n+1)$ impliziert, dass es ein Wort x der Länge n mit $\Phi_e(x) > s(|x|)$ gibt. Dies folgt jedoch unmittelbar aus der Definition von σ , l und s : Nach Definition von σ impliziert die Annahme, dass $n \geq e$ und dass es ein Wort x der Länge n gibt mit $\Phi_e(x) > s_{\sigma(e,n)}(|x|)$, und nach Definition von l impliziert die Annahme, dass $l(n) \leq \sigma(e,n)$ gilt. Mit der Definition von l und mit (6.25) folgt daher

$$s(|x|) = s_{l(|x|)}(|x|) \leq s_{\sigma(e,n)}(|x|) < \Phi_e(x).$$

□

Der Vereinigungssatz ist auf die im vorhergehenden Abschnitt eingeführten allgemeinen Komplexitätsklassen (P, NP, PSPACE, E, EXP) anwendbar. Die dort verwendeten Familien von Schrankenfunktionen S' besitzen Basen $S = \{s_e : e \geq 0\}$, die (6.25) erfüllen. Hierbei nennen wir $S = \{s_e : e \geq 0\}$ eine *Basis* von S' , falls jede Funktion s aus S auch in S' liegt und jede Funktion s' in S' von einer Funktion s in S *majorisiert* wird, d.h. $s'(n) \leq s(n)$ für fast alle Eingaben n gilt. Wie man sich leicht überlegt, gilt für jede Basis S einer Familie S'

$$C_{(\varphi, \Phi)}(S) = C_{(\varphi, \Phi)}(S').$$

Eine Basis der Familie der Polynome ist z.B. die Familie $\{e \cdot x^e + e : e \geq 0\}$. (Weitere Beispiele: s. Übungen!)

Die Schranken, die der Vereinigungssatz selbst für sehr natürliche allgemeine Komplexitätsklassen - wie die oben genannten Turingmaschinen-Klassen - liefert, sind jedoch künstlich. Man kann nämlich zeigen, wie wir im folgenden Kapitel präzisieren werden, dass die Klassen P etc. nicht durch einzelne "glatte" Schranken beschrieben werden können.