

Bmk zur Serie 7:

- Vermeidet Zirkelschlüsse oder sog. Schlangen, die sich selbst in den Schwanz beißen!

D.h. passt auf, dass ihr die zu beweisende Annahme nicht in eurem Beweis voraussetzt & verwendet.

- Zeichnet Skizzen bei den R-Reduktionen.

5.3 Die Methode der Reduktion

Def: Wir nennen eine Sprache $L \subseteq \Sigma^*$...

... entscheidbar, falls $L \in \mathcal{L}_R$ (also falls L rekursiv ist).

... berechenbar, falls $L \in \mathcal{L}_{RE}$ (also falls L rekursiv aufzählbar ist).

Def: Eine Sprache $L \subseteq \text{KodTM}$ heisst

semantisch nichttriviales Entscheidungsproblem über Turingmaschinen, falls:

(i) $\exists M_1 \text{ TM s.d. } \text{Kod}(M_1) \in L \quad (\text{somit } L \neq \emptyset)$

(ii) $\exists M_2 \text{ TM s.d. } \text{Kod}(M_2) \notin L \quad (\text{somit } L \not\subseteq \text{KodTM})$

(iii) für zwei TMs A, B mit $L(A) = L(B)$ gilt:

$\text{Kod}(A) \in L \iff \text{Kod}(B) \in L$

Satz 5.9: (Satz von Rice)

Jedes semantisch nichttriviale Entscheidungsproblem über Turingmaschinen ist unentscheidbar (d.h. $\notin \mathcal{L}_R$).

Bsp: Sei $L \subseteq \Sigma^*$ eine bel. rekursive Sprache ($L \in \mathcal{L}_R$).

Wir betrachten die Sprache

$$\text{Kod}_L := \{ \text{Kod}(M) \mid M \text{ ist eine TM und } L(M) = L \}$$

also die Kodierung aller TMs, die L akzeptieren,

und überprüfen, ob Kod_L ein semantisch nichttriviales Entscheidungsproblem über TMs ist.

(i) Da $L \in \mathcal{L}_R$ ist, $\exists M_1 \text{ TM mit } L(M_1) = L \Rightarrow \text{Kod}_L \neq \emptyset$

(ii) Offensichtlich $\exists M_2 \text{ TM mit } L(M_2) \neq L \Rightarrow \text{Kod}_L \neq \text{Kod TM}$

(iii) Für TMs A, B mit $L(A) = L(B)$ gilt $L(A) = L \iff L(B) = L$
 $\text{Kod}(A) \in \text{Kod}_L \iff \text{Kod}(B) \in \text{Kod}_L$

Kod_L ist also ein semantisch nichttriviales Entscheidungsproblem.

Satz von Rice $\Rightarrow \text{Kod}_L \notin \mathcal{L}_R$

6 Komplexitätstheorie

6.2 Komplexitätsmaße

Def: (Zeitkomplexität)

Intuitiv: Anzahl Berechnungsschritte

Sei M eine MTM, die immer hält.

Sei Σ das Eingabealphabet von M .

Sei $x \in \Sigma^*$.

Sei $D = C_1, C_2, \dots, C_k$ die Berechnung von M auf x .

Dann ist die Zeitkomplexität der Berechnung von M auf x def. durch: $\text{Time}_M(x) := k - 1$.

Die Zeitkomplexität von M ist die Fkt: $\text{Time}_M: \mathbb{N} \rightarrow \mathbb{N}$ def. durch: $\text{Time}_M(n) := \max \{ \text{Time}_M(x) \mid x \in \Sigma^n \}$.

recall: Die Konfiguration einer MTM ist wie folgt aufgebaut:

$$C = (q, x, i, \alpha_1, i_1, \alpha_2, i_2, \dots, \alpha_k, i_k) \in Q \times \Sigma^* \times \mathbb{N} \times (\Gamma^* \times \mathbb{N})^k$$

• Zustand der TM • Inhalt des Eingabebandes: $\varnothing \times \$$ • Position des Lesekopfes des Eingabebandes

• $\forall j \in \{1, \dots, k\}$: • $\varnothing \alpha_j \sqcup \sqcup$ ist der Inhalt des j -ten Arbeitsbandes

• $i_j \leq |\alpha_j|$ ist die Position, auf die der Lese-/Schreibkopf des j -ten Bandes zeigt.

Def: (Speicherkomplexität)

Sei M eine k -Band-TM ($k \in \mathbb{N} \setminus \{0\}$), die immer hält.

Sei $C = (q, x, i, \alpha_1, i_1, \alpha_2, i_2, \dots, \alpha_k, i_k)$ eine Konfiguration von M mit $0 \leq i \leq |x|+1$, $0 \leq i_j \leq |\alpha_j|$ für $j=1, \dots, k$.

Die Speicherplatzkomplexität von C ist $\text{Space}_M(C) := \max \{|\alpha_i| \mid i=1, \dots, k\}$.

Die Speicherplatzkomplexität von M auf x ist $\text{Space}_M(x) := \max \{ \text{Space}_M(C_i) \mid i=1, \dots, k \}$

Die Speicherplatzkomplexität von M ist die Fkt. $\text{Space}_M: \mathbb{N} \rightarrow \mathbb{N}$, def. durch

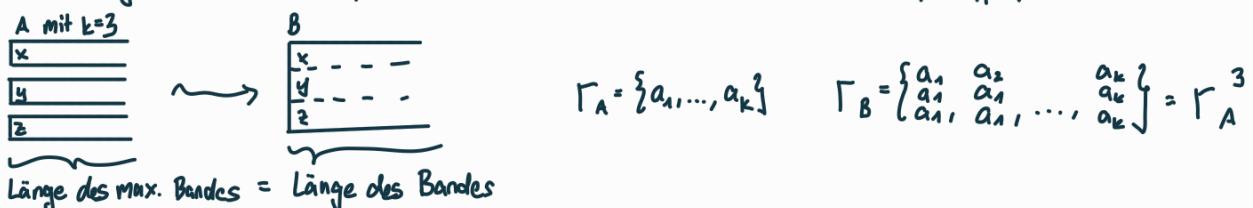
$$\text{Space}_M(n) := \max \{ \text{Space}_M(x) \mid x \in \Sigma^n \}.$$

Intuitiv: Länge des längsten Wortes, welches wir während der Berechnung speichern müssen.

LEM 6.1: Für jede k -Band-TM A ($k \in \mathbb{N} \setminus \{0\}$), die immer hält, existiert eine 1-Band-TM B , s.d.

$$\text{Space}_B(n) \leq \text{Space}_A(n)$$

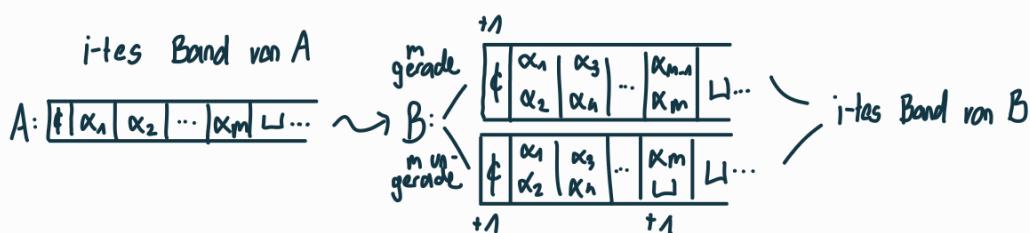
Beweisidee: Die Mächtigkeit des Arbeitsalphabets von M hat keinen Einfluss auf $\text{Space}_M(n)$.



LEM 6.2: Für jede k -Band-TM A ($k \in \mathbb{N} \setminus \{0\}$), existiert eine k -Band-TM B ,

s.d. $L(A) = L(B)$ und $\text{Space}_B(n) = \frac{\text{Space}_A(n)}{2} + 2$

Beweisidee:



Γ_B enthält alle Symbole aus $\Gamma_A \times \Gamma_A$

Def: Für jede Fkt. $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$ definieren wir:

- $g \in O(f(n))$: g wächst asymptotisch höchstens so schnell wie f .
- $g \in \Omega(f(n))$: g wächst asymptotisch mindestens so schnell wie f .
- $g \in \Theta(f(n))$: g wächst asymptotisch gleich schnell wie f .
- $g \in o(f(n))$: g wächst asymptotisch langsamer als f . ($\lim_{n \rightarrow \infty} \left(\frac{g(n)}{f(n)} \right) = 0$)

Satz 6.1 (Speedup Theorem von Blum)

Es existiert ein Entscheidungsproblem $(\Sigma_{\text{bool}}, L)$,
s.d. für jede MTM A, die $(\Sigma_{\text{bool}}, L)$ entscheidet,
eine MTM B existiert, die auch $(\Sigma_{\text{bool}}, L)$ entscheidet
s.d. für ∞ -viele $n \in \mathbb{N}$ gilt: $\text{Time}_B(n) \leq \log_2(\text{Time}_A(n))$

Intuitiv: Es gibt Probleme, bei denen sich jeder geg. Algo für die Lösung des Problems wesentlich verbessern lässt. $\Rightarrow \nexists$ kein optimaler Algo für solche Probleme.

Def: Sei L eine Sprache und $f: \mathbb{N} \rightarrow \mathbb{R}^+$ eine Fkt.

- $O(f(n))$ ist eine obere Schranke für die Zeitkomplexität von L ,
falls $\exists A$ MTM mit $L(A)=L$ und $\text{Time}_A(n) \in O(f(n))$.
- $\Omega(f(n))$ ist eine untere Schranke für die Zeitkomplexität von L ,
falls $\forall B$ MTM mit $L(B)=L$: $\text{Time}_B(n) \in \Omega(f(n))$
- Eine MTM C heißt optimal für L ,
falls $\text{Time}_C(n) \in O(f(n))$ und $\Omega(f(n))$ eine untere Schranke für die Zeitkomplexität von L ist.

Def: Eine Fkt. $s: \mathbb{N} \rightarrow \mathbb{N}$ heißt platzkonstruiert,

falls eine 1-Band-TM M existiert, s.d.

(i) $\forall n \in \mathbb{N}: \text{Space}_M(n) \leq s(n)$

(ii) für jede Eingabe 0^n mit $n \in \mathbb{N}$

generiert M das Wort $0^{s(n)}$ auf ihrem Arbeitsband und hält in q_{acc} .

Def: Eine Fkt. $t: \mathbb{N} \rightarrow \mathbb{N}$ heisst zeitkonstruierbar,

falls eine MTM A existiert, s.d.

(i) $\text{Time}_A(n) \in O(t(n))$

(ii) für jede Eingabe 0^n mit $n \in \mathbb{N}$

generiert A das Wort $0^{t(n)}$ auf dem ersten Arbeitsband und hält in q_{acc} .

Rmk: Faustregel: Für die meisten Fkt. $f: \mathbb{N} \rightarrow \mathbb{N}$ gilt:

• $f(n) \geq \log_2(n+1) \Rightarrow f$ ist platzkonstruierbar

• $f(n) \geq n \Rightarrow f$ ist zeitkonstruierbar

Aufg 1) Sei M eine 1-Band-TM, die immer hält.

Zg: Es existiert eine zu M äquivalente 2-Band-TM A,

s.d. $\forall n \in \mathbb{N}$ und eine konst c gilt:

$$\text{Time}_A(n) \leq \frac{\text{Time}_M(n)}{2} + \frac{13n}{12} + c$$

Hinweis: Die MTM A kann jeweils 12 Felder

des Eingabe- oder Arbeitsbandes von M auf einem ihrer Felder simulieren.

Musterlösung: (HS24, Serie 8, Aufgabe 24)

Die Idee hinter der Konstruktion von A ist es, jeweils 12 Schritte von M in 6 Schritten von A zu simulieren. Dafür werden jeweils 12 Felder des Arbeitsbandes von M zu einem Feld von A zusammengefasst. Die gleiche Komprimierung wird auch auf die Eingabe angewendet, hierfür nutzt A ihr zweites Arbeitsband. Wir bemerken, dass A eine konstante Anzahl von Berechnungsschritten von M in einem Schritt simulieren kann, wenn sie die von M in diesen Schritten gelesenen Symbole vorher in ihren Zuständen gespeichert hat. Wir bezahlen also für die Verkürzung der Laufzeit mit einer erheblichen Vergrößerung des Arbeitsalphabets und der Zustandsmenge. Die MTM A hat auf ihrem Eingabeband dieselbe Eingabe wie M. Um die Berechnung zu verkürzen, komprimiert A zunächst diese Eingabe auf ihrem zweiten Arbeitsband. Hierfür liest sie jeweils 12 Felder des Eingabebandes und schreibt das 12-Tupel der gelesenen Symbole auf ein Feld des zweiten Arbeitsbandes.

Dies benötigt auf einer Eingabe der Länge n offenbar $n + 1$ Schritte, weil der Kopf auf dem Eingabeband im $(n + 1)$ -ten Schritt die rechte Endmarkierung $\$$ erreicht. Danach bewegt A ihren Kopf auf dem zweiten Arbeitsband zurück an den Anfang. Weil auf diesem Arbeitsband jetzt $\lceil n/12 \rceil$ Felder belegt sind, benötigt dies $\lceil n/12 \rceil$ Schritte. Insgesamt braucht A also für dieses Preprocessing $\frac{13n}{12} + c_1$ Schritte, für eine kleine Konstante c_1 , z.B. $c_1 = 2$. Für die eigentliche Simulation von M arbeitet A in *Runden* von jeweils bis zu 6 Berechnungsschritten. In jeder Runde simuliert A dabei 12 Berechnungsschritte von M . Damit ergibt sich eine Laufzeit für die Simulation von höchstens

$$\left\lceil \frac{\text{Time}_M(n)}{12} \right\rceil \cdot 6 \leq \frac{\text{Time}_M(n)}{2} + c_2,$$

für eine kleine Konstante c_2 , z.B. $c_2 = 6$. Zu Beginn einer jeden Runde liest A die Inhalte ihrer beiden Arbeitsbänder auf dem aktuellen Feld und den beiden Nachbarfeldern links und rechts und speichert diese jeweils $3 \cdot 12 = 36$ Felder des Eingabebandes und des Arbeitsbandes von M in ihren Zuständen. Hierfür sind 4 Schritte nötig: Ein Schritt nach links, zwei Schritte nach rechts und ein Schritt zurück auf die ursprüngliche Position. Damit hat A jetzt genügend Informationen, um 12 Schritte von M in ihren Zuständen zu simulieren, denn M kann sich in 12 Schritten um höchstens 12 Positionen bewegen und A kennt nun mindestens 12 Felder links und 12 Felder rechts von der aktuellen Position von M . In den 12 simulierten Schritten kann M nur Inhalte in zwei der drei von A gelesenen Zwölferblöcken ändern. Diese Änderungen kann A nun in höchstens 2 Schritten auf ihren Bändern umsetzen: Wir beschreiben nur die Änderung des ersten Arbeitsbandes, die Änderung der Kopfposition auf dem simulierten Eingabeband kann dann analog umgesetzt werden. Im fünften Schritt der Runde ändert A den Inhalt des aktuellen Arbeitsbandfeldes entsprechend der 12 Berechnungsschritte von M und bewegt den Kopf nach links oder rechts, falls auch in dem dort simulierten Bereich des Bandes von M Änderungen nötig sind. Im sechsten Schritt führt A die Änderungen auf dem Nachbarfeld durch und kehrt gegebenenfalls zurück. Wenn auf beiden Arbeitsbändern von A nur auf der aktuellen Position Änderungen nötig sind, dann wird der sechste Schritt nicht benötigt. Insgesamt ergibt sich also für die Zeit, die A für die gesamte Berechnung benötigt, die behauptete obere Schranke.