TI Übungsstunde 9

Marcel Schmid marcesch@student.ethz.ch

11. 11.2020

---

1 Korrekturen

- Zoom: Bitte Kamera an, per Mikro oder Chat melden, etwas mehr mitmachen – es ist weird, "in die Leere" zu sprechen...
- Kleiner Unterschied: "Kanonische Ordnung über Wörter" ist eine *Ordnung*, i.e., eine mathematische Relation

⇒ "Wörter in kanonsicher Ordnung" ist eine *Liste*

- Bei Aufzählstrategien: Bitte nicht so was:

$$0, 1, -1, 2, \ldots$$

sondern formaler:

$$f(n) = \begin{cases} 0 & n = 0 \\ -i & n = 2 \cdot i, i \in \mathbb{N} \setminus \{0\} \\ i & n = 2 \cdot i + 1, i \in \mathbb{N} \end{cases}$$

- Froschaufgabe hat erstaunlich vielen Probleme bereitet – so Aufzählungsstrategien sind auch gute Prüfungsaufgaben, könnten in der Form dran kommen!
- Musterlösungen: Viele haben schon fast die Hälfte der Punkte erreicht – wenn man eine Aufgabe nicht machen will, macht sie einfach nicht da ihr eh genügend Punkte erreichen werdet – ich muss ja nicht die ML korrigieren...

  ⇒ Sich inspirieren lassen ist ok und "normal"

  ⇒ Das Schlimmste ist es, die Musterlösungen zu kopieren und kleine Details anzupassen um das Kopieren zu "vertuschen"; das bringt euch nichts und verursacht beiderseits "viel" Aufwand...(von mir aus dürft ihr auch einen Screenshot der ML reintun, ist mir prinzipiell egal...bringt euch halt einfach nicht soo viel :D)

  ⇒ Will halt auch nicht Moral predigen und Polizist spielen, aber wenn ihr das tut, dann schaut, dass ihr die Lsg. wenigstens versteht...Also Lsg. lesen, *zur Seite legen* und Beweis selber führen *ohne zu Spicken*, so bleibts wenigstens ein bisschen haften ;)

2 Theorie/Repetition

2.1 Asymptotik

(nur O erklären)

- Für jede Funktion $f : \mathbb{N} \to \mathbb{R}^+$ definieren wir

$$O(f(n)) = \{r : \mathbb{N} \to \mathbb{R}^+ \mid \exists n_0 \in \mathbb{N}, \exists c \in \mathbb{N} \text{ s.t. for all } n \geq n_0 : r(n) \leq c \cdot f(n)\}$$

- Analog:

$$\Omega(f(n)) = \{r : \mathbb{N} \to \mathbb{R}^+ \mid \exists n_0 \in \mathbb{N}, \exists d \in \mathbb{N} \text{ s.t. for all } n \geq n_0 : r(n) \leq \frac{1}{d} \cdot f(n)\}$$

- Schliesslich

$$\Theta(h(n)) = O(h(n)) \cap \Omega(h(n))$$

2.2 Time, Space

(Mit Bildern/Diagrammen erklären)

- Zeitkomplexität der Berechnung von M auf x ist $\text{Time}_M(x) = k - 1$, wobei:
  - M: eine MTM, die immer hält
  - x eine Eingabe von M (über Alphabet Σ)
  - $D = C_1, C_2, \ldots, C_k$ die Berechnung von M auf x
- Die Zeitkomplexität von M ist die Funktion $\text{Time}_M(n) : \mathbb{N} \to \mathbb{N}$ wobei
$$\text{Time}_M(n) = \max\{\text{Time}_M(x) \mid x \in \Sigma^n\}$$
also das maximum über alle Eingaben x einer Länge n
- Speicherplatzkomplexität: Sei $C = (q, x, \alpha_1, i_1, \ldots, \alpha_k, i_k)$ für $0 \leq i \leq |x| + 1$ und $0 \leq i_j \leq |\alpha_j|$ für $j = 1, \ldots, k$. Dann:
  - $\text{Space}_M(C) = \max\{|\alpha_i| \mid i = 1, \ldots, k\}$ (die Länge des längsten Bandes)
  - Sei $C_1, \ldots, C_l$ die Berechnung von M auf x:
$$\text{Space}_M(x) = \max\{\text{Space}_M(C_i) \mid i = 1, \ldots, l\}$$
  - Die Speicherplatzkomplexität von M ist
$$\text{Space}_M(n) = \max\{\text{Space}_M(x) \mid x \in \Sigma^n\}$$
- Für jede k-Band TM A exisitert eine äquivalente 1-Band TM so dass
$$\text{Space}_B(n) \leq \text{Space}_A(n)$$
$\Rightarrow$ wir können das Alphabet beliebig gross/mächtig machen!

2.3 Zeitkonstruierbarkeit

Aufgabe: Ist $2^{\lceil\sqrt{n}\rceil}$ zeitkonstruierbar?

Ja: wir geben TM A an, welche wie folgt funktioniert: (könnte man auch noch formaler machen, aber ja...)

1. Für Eingabe 0^n liest A die Eingabe in Zeit n und berechnet den Wert $2^{\lceil\sqrt{n}\rceil}$ in Zeit $O(2^{\sqrt{n}})$ auf dem ersten Arbeitsband.
2. Dann geht initialisiert A das zweite Band als 0, schreib eine "0" auf das Eingabeband und erhöht den Zähler auf B2 um 1.
3. Sobald der Inhalt des zweiten Bandes demjendigen des ersten entspricht terminiert A.
4. Offenbar braucht A Zeit $O(2^{\sqrt{n}})$ (sicherlich nicht mehr zur Berechnung und Speichern von $2^{\lceil\sqrt{n}\rceil}$ und der Loop terminiert nach $2^{\lceil\sqrt{n}\rceil}$ vielen Schritten).
5. Aus unserer Konstruktion folgt ausserdem, dass wir eine MTM A haben, welche die Bedingungen der Definition erfüllt, somit ist f zeitkonstruierbar.

2.4 Deterministische vs. Nichtdeterministische Zeitmasse

Wichtige Definitionen:

$$\begin{aligned}
\textbf{TIME}(f) &= \{L(B) \mid B \text{ ist MTM mit } \text{Time}_B(n) \in O(f(n))\} \\
\textbf{SPACE}(g) &= \{L(A) \mid A \text{ ist MTM mit } \text{Space}_A(n) \in O(g(n))\} \\
\textbf{P} &= \bigcup_{c \in \mathbb{N}} \text{TIME}(n^c) \\
\textbf{EXPTIME} &= \bigcup_{d \in \mathbb{N}} \text{TIME}(2^{n^d})
\end{aligned}$$

NTIME, NSPACE, NP analog. Wichtig:

$$\textbf{NLOG} \subseteq \textbf{P} \subseteq \textbf{NP} \subseteq \textbf{PSPACE} \subseteq \textbf{EXPTIME}$$

2.5 Verifizierer

(intuition – e.g. optimale Lsg. = Verifizierer etc.) Sei $L \subseteq \Sigma^*$ eine Sprache und sei $p : \mathbb{N} \to \mathbb{N}$ eine Funktion. Wir sagen, dass eine MTM A ein p-Verifizierer für L ist, falls A mit folgenden Eigenschaften auf allen Eingaben aus $\Sigma^* \times (\Sigma_{\text{bool}})^*$ arbeitet:

1. $\text{Time}_A(w, x) \leq p(|w|)$ für jede Eingabe $(w, x) \in \Sigma^*(\Sigma_{\text{bool}})^*$
2. Für jedes w in L gibt es ein $x \in (\Sigma_{\text{bool}})^*$ so dass $|x| \leq p(|w|)$ und $(w, x) \in L(A)$

$\Rightarrow VP = \{V(A) \mid A \text{ ist poly-time Verifizierer}\}$
$\Rightarrow VP = NP.$

3 Übungen

3.1 "Eingabe-Verwerfen-Trick"

Sei $L_{\text{all}} = \{\text{Kod}(M) \mid L(M) = \Sigma^*\}$. Zeige $L_u \leq_{EE} L_{\text{all}}$:

1. Wir beschreiben eine TM A, welche eine Eingabe für L_U zu einer Eingabe für L_{all} umwandelt und wie folgt auf einer Eingabe x arbeitet:
2. Falls x nicht von der Form $\text{Kod}(M)\#w$ ist, so gibt A $\text{Kod}(B_\emptyset)$ aus, wobei $B_\emptyset$ eine (triviale) TM ist, welche keine Eingabe akzeptiert.
3. Andernfalls gilt $x = \text{Kod}(M)\#w$ für eine TM M und $w \in \Sigma^*$. Dann gibt A $\text{Kod}(B_x) = \text{Kod}(B_{\text{Kod}(M)\#w})$ aus, wobei $B_x = B_{\text{Kod}(M)\#w}$ wie folgt arbeitet:
   (a) B_x erhält die Eingabe $y \in \Sigma^*$
   (b) B_x löscht die Eingabe y und schreibt $x = \text{Kod}(M)\#w$ aufs Band (x ist so quasi durch "Hard-coding" in B drin, x ist **keine** Eingabe!). Dann simuliert B_x die Arbeit von M auf w.
   (c) B akzeptiert die Eingabe y gdw. M die Eingabe w akzeptiert.
   $\Rightarrow$ Intuition: Da B_x im zweiten Schritt seine Eingabe gelöscht hat, gilt das für jedes beliebige $y \Rightarrow$ daher akzeptiert B_x jedes $y \in \Sigma^*$ gdw $w \in L(M)$.
4. Korrektheit: wir zeigen $x \in L_U \iff A(x) \in L_{\text{all}}$:
   "$\Rightarrow$": Sei $x \in L_U$. Dann ist x von der Form $\text{Kod}(M)\#w$ und $w \in L(M)$. Aus unserer Konstruktion folgt dann, dass A $\text{Kod}(B_x) = \text{Kod}(B_{\text{Kod}(M)\#w})$ ausgibt. Da $w \in L(M)$ ist, akzeptiert auch B seine Eingabe y (folgt aus Konstruktion).
   $\Rightarrow$ Für beliebiges $y \in \Sigma^*$ akzeptiert B_x die Eingabe y.
   $\Rightarrow \forall y \in \Sigma^* : y \in L(B)$
   $\Rightarrow L(B) = \Sigma^*$.
5. "$\Leftarrow$": Sei $x \notin L_U$. Dann gibts zwei Fälle:
   - Entweder ist x von der falschen Form. Dann gibt A $\text{Kod}(B_\emptyset)$ aus und somit ist $A(x) \notin L_{\text{all}}$.
   - Oder $w \notin L(M)$. Sei $y \in \Sigma^*$ eine beliebige Eingabe für B_x. Da $w \notin L(M)$, verwirft B_x entweder die Eingabe y in endlicher Zeit, oder B_x rechnet unendlich. So oder so gilt $y \notin L(B)$ und somit ist $L(B) \neq \Sigma^*$.
     $\Rightarrow A(x) \notin L_{\text{all}}$
6. Bemerkung: Für L_{all} könnte man auch gut den Satz von Rice verwenden, um viel schneller zu zeigen, dass $L_{\text{all}} \notin \mathcal{L}_R$.

3.2 S19, 23

Sei M eine 1-Band TM, die immer hält. Zeige, dass es eine dazu äquivalente 2-Band TM A gibt, so dass für c konst. und für alle n gilt dass

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

1. Die Hauptidee ist es, auszunutzen, dass die Zustände Q und das Arbetisalphabet beliebig mächtig sein können. Wir können so nämlich mehrere Schritte auf einmal simulieren. Das schwierige ist es nun, das korrekt zu machen. Wir können jeweils 12 Schritte von M in 6 Schritten von A simulieren wie folgt:

2. **Preprocessing**: zuerst schreiben wir jeweils 12 Symbole auf dem Eingabeband als ein 12-Tupel wie folgt: A liest jeweils 12 Symbole auf dem Eingabeband und schreibt das entsprechende 12-Tupel auf das erste Arbeitsband.

   Nach n gelesenen Symbolen gibt es $\lceil \frac{n}{12} \rceil$ viele Zeichen (i.e. 12-Tupel) auf dem Arbeitsband. Damit die Simulation korrekt verläuft, muss A noch den Kopf *auf dem Arbeitsband* ganz nach links zurückbewegen – dafür brauchts $\lceil n/12 \rceil$ viele Schritte.

   Also hat A bisher $n+1$ (Lesen von Eingabe) + $\lceil n/12 \rceil$ (Bewege Kopf zurück nach links) viele Schritte, also $\frac{13n}{12} + c_1$ gesamthaft.

3. **Simulation**: A arbeitet in Runden, wobei jede Runde bis zu 6 Schritte braucht und dabei 12 Schritte von M simuliert werden. A liest dabei die gesamte "lokale Konfiguration" (i.e., alles, was zur Berechnung der 12 Schritte nötig ist) und speichert die in ihren Zuständen:

4. **Beginn einer Runde**: Zu Beginn muss sich A zuerst einmal "einen Überblick verschaffen" über die aktuellen Inhalte. Wenn der Lesekopf momentan auf dem 12-Tupel t_i ist, so kann in den 12 simulierten Berechnungsschritten auch der Bandinhalt von den Tupeln t_{i-1}, t_{i+1} beeinflusst werden. Deswegen liest A diese 12-Tupel ein:

   A bewegt den Kopf von t_i um eins nach links auf t_{i_1}, speichert den aktuellen Inhalt als Zustand. Dann geht A nach t_{i+1}, liest wiederum den Inhalt und merkt sich das als Zustand. Schliesslich muss A den Kopf wieder auf t_i bewegen, da die Simulation von dort aus startet.

   Alles in allem braucht A für das also 6 Schritte: $l \to r \to r \to l$.

5. **Berechnung**: Sobald A den Inhalt von t_{i-1}, t_i, t_{i+1} kennt (was 36 Feldern in M entspricht), kann A basierend darauf die 12 Schritte gleichzeitig berechnen. Bemerke: in diesen 12 Schritten kann es sein, dass nicht nur die Inhalte von t_i, sondern auch von t_{i+1} *oder* t_{i-1} verändert werden (aber nicht beide!). Daher muss A am Ende dieser Runde das Resultat in t_i schreiben und den Kopf allenfalls auf $t_{i\pm1}$ bewegen, das Tupel dort modifizieren und zurück zu t_i kommen. Das geht in 2 Schritten.

6. $\Rightarrow$ alles in allem bruacht A also 6 Schritte, um 12 Schritte von M auszuführen. Damit erhalten wir eine Laufzeit von $\leq \frac{\text{Time}_M(n)}{2} + c_2$ (Rundungen etc. sind in c_2) und somit alles in allem:

$$\text{Time}_A(n) \leq \frac{\text{Time}_M(n)}{2} + c_2 + \frac{13n}{12} + c_1$$

4 Neue Serie

- Aufgepasst: R-Reduktionen eignen sich *nicht* zum Beweisen, dass eine Sprache L nicht in $\mathcal{L}_{RE}$ ist!

  $\Rightarrow$ stattdessen "Trick" von letztem Mal verwenden (allenfalls auch EE-Reduktion mit Proof Outline dass mit EE-Reduktion zu so Beweisen taugt)