Grundzüge der Theoretischen Informatik, WS 21/22: Musterlösung zum 11. Präsenzblatt

Julian Dörfler

Aufgabe P11.1 (Beweismethoden)
Gegeben eine Menge $A_{\mathbb{N}} \subseteq \mathbb{N}$ oder $A_{\Sigma} \subseteq \Sigma^{\star}$ und eine partielle Funktion $f : \mathbb{N} \to \mathbb{N}$. Welche Möglichkeiten gibt es die folgenden Aussagen zu beweisen?

(a) $A_{\Sigma} \in \mathsf{REG}$

(b) $A_{\Sigma} \notin \mathsf{REG}$

(c) $A_{\mathbb{N}} \in \mathsf{REC}$

(d) $A_{\mathbb{N}} \notin \mathsf{REC}$

(e) $f \in \mathsf{R}$

(f) $f \notin \mathsf{R}$

(g) $A_{\mathbb{N}} \in \mathsf{RE}$

(h) $A_{\mathbb{N}} \notin \mathsf{RE}$

(i) $A_{\mathbb{N}} \in \mathsf{co\text{-}RE}$

(j) $A_{\mathbb{N}} \notin \mathsf{co\text{-}RE}$

(k) $A_{\Sigma} \in \mathsf{P}$

(l) $A_{\Sigma} \in \mathsf{NP}$

(m) A_{Σ} ist NP-schwer

(n) A_{Σ} ist NP-vollständig

Lösung P11.1 (Beweismethoden)

(a) Wir zeigen $A_{\Sigma} \in \mathsf{REG}$ indem wir entweder einen regulären Ausdruck oder einen DEA oder NEA konstruieren, der die Sprache erkennt. Alternativ für etwas abstraktere Sprachen können wir auch mithilfe des Myhill-Nerode-Theorems eine endliche Anzahl an Myhill-Nerode-Äquivalenzklassen nachweisen sowie die Abschlusseigenschaften der regulären Sprachen verwenden.

Für unäre Sprachen können wir dies auch durch zeigen der letztendlich-periodisch Eigenschaft mit Aufgabe P2.3 beweisen.

(b) Wir zeigen $A_\Sigma \notin \mathsf{REG}$ mithilfe des Pumping-Lemmas oder weisen mithilfe des Myhill-Nerode-Theorems eine unendliche Anzahl an Myhill-Nerode-Äquivalenzklassen nach. Wir können die Sprachen vorher durch Abschlusseigenschaften der regulären Sprachen (z.B. Schnitt, Vereinigung, Komplement, Homomorphismen, Transduktoren) vorbearbeiten um den Beweis möglich oder leichter zu machen oder sogar direkt auf schon bekannte Sprachen zurückzufuhren.

Für unäre Sprachen können wir dies auch durch widerlegen der letztendlich-periodisch Eigenschaft mit Aufgabe P2.3 beweisen.

(c) Wir zeigen $A_\mathbb{N} \in \mathsf{REC}$ indem wir eine *Beschreibung* eines `WHILE`-Programmes angeben, dass $\chi_{A_\mathbb{N}}$ berechnet.

(d) Wir zeigen $A_\mathbb{N} \notin \mathsf{REC}$ entweder mithilfe des Satzes von Rice (für nicht-triviale Indexmengen) oder mithilfe einer Reduktion. In seltenen Fällen können wir dies auch Zeigen indem wir die Existenz eines `WHILE`-Programmes zu einem Widerspruch führen, z.B. per Diagonalisierung.

(e) Wir zeigen $f \in \mathsf{R}$ indem wir eine *Beschreibung* eines `WHILE`-Programmes angeben, dass f berechnet.

(f) Wir zeigen $f \notin \mathsf{R}$ indem wir die Existenz eines `WHILE`-Programmes zu einem Widerspruch führen, z.B. per Diagonalisierung oder via einer Sprache die nicht in REC ist.

(g) Wir zeigen $A_\mathbb{N} \in \mathsf{RE}$ indem wir eine *Beschreibung* eines `WHILE`-Programmes angeben, dass $\chi'_{A_\mathbb{N}}$ berechnet.

(h) Wir zeigen $A_\mathbb{N} \notin \mathsf{RE}$ entweder mithilfe des expliziten Satzes von Rice aus Aufgabe P8.1 (für nicht-triviale Indexmengen, die Ω enthalten) oder mithilfe einer Reduktion von $\overline{H_0}$.

Alternativ: Wenn wir schon gezeigt haben, dass $A_\mathbb{N} \in \mathsf{co\text{-}RE}$, aber $A_\mathbb{N} \notin \mathsf{REC}$, dann kann $A_\mathbb{N} \in \mathsf{RE}$ schon nicht gelten.

(i) Wir zeigen $A_\mathbb{N} \in \mathsf{co\text{-}RE}$ indem wir eine *Beschreibung* eines `WHILE`-Programmes angeben, dass $\chi'_{\overline{A_\mathbb{N}}}$ berechnet.

(j) Wir zeigen $A_\mathbb{N} \notin \mathsf{co\text{-}RE}$ entweder mithilfe der expliziten Satzes von Rice aus Aufgabe P8.1 (für nicht-triviale Indexmengen, die Ω nicht enthalten) oder mithilfe einer Reduktion von H_0.

Alternativ: Wenn wir schon gezeigt haben, dass $A_\mathbb{N} \in \mathsf{RE}$, aber $A_\mathbb{N} \notin \mathsf{REC}$, dann kann $A_\mathbb{N} \in \mathsf{co\text{-}RE}$ schon nicht gelten.

(k) Wir zeigen $A_\Sigma \in \mathsf{P}$ indem wir eine *Beschreibung* einer Polynomialzeit DTM angeben, die A_Σ entscheidet.

(l) Wir zeigen $A_\Sigma \in \mathsf{NP}$ indem wir entweder eine *Beschreibung* einer Polynomialzeit NTM angeben, die A_Σ entscheidet oder indem wir einen Polynomialzeitverifizierer für A_Σ angeben.

(m) Wir zeigen, dass A_Σ NP-schwer ist durch eine Polynomialzeitreduktion von einem Problem für dass wir schon bewiesen haben, dass dieses NP-schwer ist.

(n) Wir zeigen, dass A_Σ NP-vollständig ist, indem wir beweisen dass $A_\Sigma \in \mathsf{NP}$ und dass A_Σ NP-schwer ist.