Theoretische Informatik 2

Sommersemester 2018

Elizaveta Kovalevskaya
Georg Schnitger
Philipp Tertel

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Institut für Informatik
AG Theoretische Informatik

Übungsblatt 8

Ausgabe: 05.06.18
Abgabe: 12.06.18

10 Bonuspunkte können erworben werden.

Aufgabe 8.1 *Alternierende Berechnungen* (4+4+2 Punkte)

Wir führen alternierende Turingmaschinen (ATMs) als Verallgemeinerung von nichtdeterministischen Turingmaschinen ein. Eine ATM wird durch ein Tupel

$$A = (Q_\exists, Q_\forall, \Sigma, \Gamma, \delta, q_0, F_{\text{ja}}, F_{\text{nein}})$$

beschrieben, wobei $Q_\exists$ und $Q_\forall$ disjunkte Mengen *existenzieller* bzw. *universeller Zustände* mit $Q := Q_\exists \cup Q_\forall$ sind, Σ das *Eingabealphabet* und Γ (mit $\Sigma \subseteq \Gamma$) das *Bandalphabet* ist, $q_0 \in Q$ der *Anfangszustand*,

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\text{links, rechts, bleib}\})$$

die Übergangsfunktion sowie F_{ja} und F_{nein} die disjunkten Mengen der *akzeptierenden* bzw. *verwerfenden Haltezustände*. Für alle $q \in F_{\text{ja}} \cup F_{\text{nein}}$ und alle $\gamma \in \Gamma$ ist $\delta(q, \gamma) = \emptyset$.

Wie üblich benutzen wir Konfigurationen uqv – mit $u, v \in \Gamma^*$ und $q \in Q$ –, um auszudrücken, dass sich A im Zustand q befindet, uv der Inhalt des Bands ist und der Lese/Schreibkopf den ersten Buchstaben von v liest. Für eine Eingabe w ist q_0w die Anfangskonfiguration.

ATMs unterscheiden sich in ihrem Akzeptanzverhalten von nichtdeterministischen Turingmaschinen. Wir geben eine rekursive Definition des Begriffs einer *akzeptierenden* bzw. *verwerfenden Konfiguration* an und beginnen mit Endkonfigurationen, also mit Konfigurationen uqv für $q \in F_{\text{ja}} \cup F_{\text{nein}}$.

Basisregel: Eine Konfiguration uqv ist akzeptierend, wenn $q \in F_{\text{ja}}$, bzw. verwerfend, wenn $q \in F_{\text{nein}}$.

Rekursive Regeln:

- Für $q \in Q_\exists \setminus (F_{\text{ja}} \cup F_{\text{nein}})$ ist eine Konfiguration uqv genau dann akzeptierend, wenn *mindestens eine* direkte Nachfolgekonfiguration akzeptierend ist.
- Für $q \in Q_\forall \setminus (F_{\text{ja}} \cup F_{\text{nein}})$ ist eine Konfiguration uqv genau dann akzeptierend, wenn *alle* direkten Nachfolgekonfiguration akzeptierend sind.

Schließlich definieren wir $L(A) := \{\, w \in \Sigma^* \; : \; q_0w \text{ ist eine akzeptierende Konfiguration}\}$ als die von A akzeptierte Sprache.

a) Geben Sie eine informelle Beschreibung einer ATM, die die Sprache QBF in polynomieller Zeit erkennt.

b) Sei AP die Klasse aller Sprachen, die von ATMs in polynomieller Zeit erkannt werden können. Zeigen Sie mit Hilfe von a), dass

$$\textbf{PSPACE} \subseteq \textbf{AP}.$$

c) Zeigen Sie mit Hilfe von b), dass

$$\textbf{PSPACE} = \textbf{AP}.$$

Aufgabe 8.2 *Inäquivalenz von NFAs* (2+6 Punkte)

Wenn überprüft werden soll, ob $L(A_1) \neq L(A_2)$ für zwei DFAs A_1 und A_2 gilt, ob also A_1 und A_2 inäquivalent sind, dann gelingt dies in Zeit $O(n \log_2 n)$, wenn n das Maximum der Zustandszahlen von A_1 und A_2 ist: Man bestimmt den Äquivalenzklassenautomat A_1' von A_1 bzw. A_2' von A_2 und überprüft, ob A_1' und A_2' nicht isomorph sind. Das Inäquivalenzproblem für NFAs ist ungleich schwieriger.

Wir beginnen mit dem Nicht-Universalitätsproblem „$L(A) \overset{?}{\neq} \Sigma^*$" für einen NFA $A = (\Sigma, Q, \delta, q_0, F)$.

a) Zeigen Sie: Wenn $L(A) \neq \Sigma^*$, dann verwirft A ein Wort $w \in \Sigma^*$ der Länge höchstens $2^{|Q|}$.

b) Entwerfen Sie einen nichtdeterministischen Algorithmus, der die Sprache der Nicht-Universalität mit linearem Speicherplatz akzeptiert: Ihr Algorithmus muss einen NFA also genau dann akzeptieren, wenn $L(A) \neq \Sigma^*$ gilt.

   Kommentar: In der Vorlesung wird gezeigt, dass die Nicht-Universalität **PSPACE**-hart ist und wir können folgern, dass die Nicht-Universalität **PSPACE**-vollständig ist. Mit ähnlichen Argumenten kann auch gezeigt werden, dass das Inäquivalenzproblem für NFAs zur Klasse **PSPACE** gehört. Da das Inäquivalenzproblem aber mindestens so schwierig wie die Nicht-Universalität ist, ist auch das Inäquivalenzproblem **PSPACE**-vollständig.

Aufgabe 8.3 *NL, NP und PSPACE* (3+2+3 Punkte)

a) Zeigen Sie, dass **NL** eine echte Teilmenge von **PSPACE** ist.

   Kommentar: Es ist nicht bekannt, ob **DL** oder **NL** echte Teilmengen von **NP** sind.

(b) **PSPACE**-vollständige Sprachen sind mindestens so schwierig wie **NP**-vollständige Sprachen, in aller Wahrscheinlichkeit aber noch viel schwieriger. Warum?

   i) Zeigen Sie: Wenn die Sprache L **PSPACE**-hart ist, dann ist L auch **NP**-hart.

   ii) Zeigen Sie: Wenn eine **PSPACE**-harte Sprache in **NP** liegt, dann folgt **NP** = **PSPACE**.

Aufgabe 8.4 *Das Katze-Maus-Spiel: Die exakte Komplexität* (8 Punkte)

Im Katze-Maus-Spiel ist ein gerichteter Graph $G = (V, E)$ gegeben sowie Positionen $k \in V$ für die *Katze*, $m \in V$ für die *Maus* und $\ell \in V$ für das Maus*loch*. Es gelte $k \neq \ell$. Für jeden Knoten $v \in V$ sei $\mathcal{N}(v) := \{v\} \cup \big\{u : \{u, v\} \in E\big\}$ die Menge der Nachbarn von v (inkl. v).

Die beiden Spieler ziehen abwechselnd. Die Katze beginnt, wählt einen Nachbarn $k' \in \mathcal{N}(k) \backslash \{\ell\}$ und aktualisiert ihre Position (setze $k := k'$). Danach ist die Maus am Zug, wählt einen Nachbarn $m' \in \mathcal{N}(m)$ und aktualisiert ebenfalls ihre Position (setze $m := m'$). Gilt irgendwann $k = m$, dann endet das Spiel und die Katze gewinnt. Gilt irgendwann $m = \ell$, endet das Spiel ebenfalls und die Maus gewinnt.

Wir definieren die Sprache

$$\text{KM} := \{(G, k, m, l) \ : \ k \neq l \text{ und die Katze hat eine Gewinnstrategie}\}.$$

Unter der Annahme $\mathbf{P} \neq \mathbf{NP} \neq \mathbf{PSPACE}$: Gehört KM zu **P**, ist KM **NP**-vollständig oder sogar **PSPACE**-vollständig? Begründen Sie Ihre Antwort.

Hinweis: Für welche „Konfigurationen" (k, m, l) ist *sofort* klar, dass die Katze eine Gewinnstrategie hat?