Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 4

Vorlesung Theoretische Grundlagen der Informatik im WS 18/19

Ausgabe 4. Dezember 2018
Abgabe 18. Dezember 2018, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1

(3 Punkte)

Die in der Vorlesung definierte nichtdeterministische Turingmaschine wird auch als RV-NTM bezeichnet (Raten/Verifizieren). Eine andere Möglichkeit ist es, NTMs analog zu NEAs zu definieren (vgl. Wegener und Übung):

Eine solche alternative nichtdeterministische Turingmaschine (A-NTM) ist definiert wie eine TM, nur ist die Übergangsfunktion δ durch eine zweistellige Relation $\subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R, N\})$ ersetzt, die wir ebenfalls δ nennen.

Die Arbeitsweise einer A-NTM ist die folgende. Wenn die A-NTM im Zustand $q \in Q$ das Zeichen $a \in \Gamma$ liest, ist für jedes $((q, a), (q', a', d)) \in \delta$ der Rechenschritt möglich, den eine DTM für $\delta(q, a) = (q', a', d)$ durchführt. Falls kein Rechenschritt möglich ist, stoppt die A-NTM. Für eine feste Eingabe x können offensichtlich viele Rechenwege möglich sein.

Eine A-NTM $\mathcal{M}$ akzeptiert eine Eingabe x, falls es mindestens einen Rechenweg von $\mathcal{M}$ gibt, der in einen akzeptierenden Endzustand führt. Die von $\mathcal{M}$ erkannte Sprache $L = L(\mathcal{M})$ besteht aus allen Worten, die $\mathcal{M}$ akzeptiert.

Die Rechenzeit für eine Eingabe $x \in L(\mathcal{M})$ ist gleich der Anzahl der Rechenschritte auf einem kürzesten akzeptierenden Rechenweg. Die Zeitkomplexität $T_{\mathcal{M}}(n)$ ist das Maximum der Rechenzeiten für alle Eingaben aus $L(\mathcal{M})$ der Länge n, und 1 falls es keine solche Eingabe gibt.

Die Klasse ANP ist die Klasse aller Probleme, die von einer solchen A-NTM in polynomieller Zeit entschieden werden können. In der Übung wurde/wird[1] gezeigt, dass $\mathsf{ANP} \subseteq \mathsf{NP}$ gilt. Zeigen Sie: $\mathsf{NP} \subseteq \mathsf{ANP}$.

Aufgabe 2

(3 + 4 + 2 = 9 Punkte)

In der Vorlesung wurden Turingmaschinen mit einem eindimensionalen, in beide Richtungen unbeschränkten Band betrachtet. Der Index eines Feldes kann hier durch eine ganze Zahl $i \in \mathbb{Z}$ repräsentiert werden. Wir nennen diese Turingmaschinen daher im Folgenden $\mathbb{Z}$-TM. In dieser Aufgabe betrachten wir einige Turingmaschinen-Modelle mit anderen Ausprägungen des Bandes:

[1] am 6.12.2018

- $\mathbb{N}_0$-TM haben ein eindimensionales, in eine Richtung beschränktes Band, d.h. links von der Startposition 0 gibt es keine weiteren Felder.
- $\mathbb{Z}^2$-TM haben ein zweidimensionales, in jede Richtung unbeschränktes Band. Die Felder können hier durch zweistellige Koordinaten adressiert werden, z.B. ist die Startposition des Kopfes $(0, 0)$.
- $\mathbb{N}_0^2$-TM haben ein zweidimensionales, in beiden Dimensionen einseitig beschränktes Band, d.h. Felder mit negativen Koordinaten existieren nicht.

Gemäß der Church'schen These sollten diese drei Modelle genauso mächtig wie $\mathbb{Z}$-TMs sein. Dies sollen Sie in dieser Aufgabe zeigen, indem Sie die entsprechenden (nichttrivialen) Simulationen angeben.

(a) Beschreiben Sie, wie eine $\mathbb{Z}$-TM von einer $\mathbb{N}_0$-TM simuliert werden kann.

(b) Beschreiben Sie, wie eine $\mathbb{N}_0^2$-TM von einer $\mathbb{Z}$-TM simuliert werden kann.

(c) Beschreiben Sie, wie eine $\mathbb{Z}^2$-TM von einer $\mathbb{Z}$-TM simuliert werden kann.

Sie dürfen dabei die in der Übung bewiesene Tatsache verwenden, dass k-Band-TM genauso mächtig sind wie 1-Band-TM. Ihre simulierende TM darf also eine beliebige (aber konstante!) Anzahl von Bändern verwenden.

Aufgabe 3

(3 + 2 + 2 = 7 Punkte)

(a) Geben Sie eine polynomielle Transformation von 3COLOR zu SAT an. Geben Sie dazu eine Vorschrift an, wie ein Graph $G = (V, E)$ in eine Menge von Klauseln überführt werden kann, die genau dann erfüllbar ist, wenn G dreifärbbar ist. Beschreiben Sie die Bedeutung der Variablen und Klauseln. Zeigen Sie, dass die Anzahl der Variablen und Klauseln sowie die Länge der Klauseln polynomiell in der Eingabegröße der 3COLOR-Instanz ist.

(b) Geben Sie eine *lineare* Transformation von SUBSET SUM zu KNAPSACK an.

(c) Gegeben seien zwei Probleme A und B und eine polynomielle Transformation von A nach B. Es sind keine weiteren Eigenschaften bekannt. Welche der folgenden Aussagen folgen *allein* aus der Existenz der polynomiellen Transformation? Begründen Sie!

(i) B ist NP-vollständig.

(ii) Wenn A NP-schwer ist, dann ist B NP-schwer.

(iii) Wenn B NP-vollständig ist, dann ist A NP-schwer.

(iv) Wenn A NP-schwer ist, dann ist B NP-vollständig.

Aufgabe 4

(3 Punkte)

Bezeichne mit 42-COLOR das Problem zu entscheiden, ob sich die Knoten eines Graphen mit höchstens 42 Farben färben lassen, so dass keine zwei adjazenten Knoten dieselbe Farbe erhalten. Zeigen Sie die NP-Vollständigkeit von 42-COLOR, indem sie von 3-COLOR reduzieren.

Aufgabe 5 (2 Punkte)

In Vorlesung und Übung wurde gezeigt, wie eine beliebige Turingmaschine $\mathcal{M}$ durch ein Wort $\langle \mathcal{M} \rangle \in \{0,1\}^\star$ kodiert werden kann. Umgekehrt haben wir (ohne genauere Erläuterungen) gefordert, dass jedes Wort aus $\{0,1\}^\star$ eine Turingmaschine kodieren möge. Anders als in der Vorlesung lassen wir nun die Vereinbarung, dass ein Wort, das keine Turingmaschine im Sinne der Gödelnummer beschreibt, eine Turingmaschine kodiert, die die Sprache $\emptyset$ akzeptiert, fallen. Stattdessen möchten wir (zur späteren Verwendung in Aufgabe 6) jede Turingmaschine durch unendlich viele Wörter aus $\{0,1\}^\star$ kodieren.

Beschreiben Sie eine Dekodierungsfunktion $\langle \cdot \rangle^{-1}$, die jedem Wort aus $\{0,1\}^\star$ eine Turingmaschine zuordnet und die die genannten Forderungen erfüllt.

Aufgabe 6 (3 + 2 + 4 = 9 Punkte)

Die Algorithmik beschäftigt sich intensiv mit dem Entwurf von Algorithmen mit möglichst geringer Laufzeit. Aber können z.B. innerhalb quadratischer Laufzeit wirklich mehr Sprachen entschieden werden als in linearer Zeit? Ziel dieser Aufgabe ist es, zu beweisen, dass dies tatsächlich so ist, dass also $\mathsf{DTIME}(n) \subsetneq \mathsf{DTIME}(n^2)$ gilt. Zur Erinnerung:

> $\mathsf{DTIME}(n^k)$ bezeichnet die Menge der Sprachen, die von einer deterministischen Turingmaschine in $\mathcal{O}(n^k)$ Laufzeit entschieden werden können.

Wir verwenden ein Diagonalargument, das ganz ähnlich ist zu dem Beweis, dass die Diagonalsprache unentscheidbar ist. Dazu definieren wir eine Sprache, die in quadratischer Laufzeit entschieden werden kann und zeigen dann, dass diese Sprache von keiner Turingmaschine in Linearzeit entschieden werden kann.

Betrachten Sie dazu die Turingmaschine D. Bei Eingabe von w läuft D für exakt $|w|^2$ Schritte. Insbesondere gilt also $L(D) \in \mathsf{DTIME}(n^2)$. Dabei simuliert D die Turingmaschine T_w auf Eingabe w. Sei m die Anzahl der Ausführungsschritte von T_w, die durch D in $|w|^2$ Schritten simuliert werden können. Um m Schritte einer Turingmaschine zu simulieren, braucht man im Allgemeinen $\Theta(m \log m)$ Schritte. Wir gehen davon aus, dass D $cm \log m$ Schritte braucht (mit einer Konstante c), um m Schritte von T_w zu simulieren. Es gilt also $cm \log m = |w|^2$. Falls T_w innerhalb dieser m Schritte terminiert, so akzeptiert D die Eingabe w genau dann, wenn T_w die Eingabe w ablehnt. Ansonsten lehnt D die Eingabe w ab.

Wir möchten nun zeigen, dass es keine Turingmaschine gibt, die $L(D)$ in Linearzeit entscheidet. Um Aussagen über alle Turingmaschinen zu machen, betrachten wir ein beliebiges Wort w und interpretieren dieses als kodierte Turingmaschine T_w. Auch dieses Vorgehen kennen Sie schon vom Beweis der Unentscheidbarkeit der Diagonalsprache.

(a) Gehen Sie zunächst davon aus, dass die Simulation von T_w bei Eingabe von w stets innerhalb der geforderten Anzahl an Schritten terminiert. Zeigen Sie, dass dann $L(T_w) \neq L(D)$ gilt.

(b) Sei T_w eine Turingmaschine, die $L(D)$ in Linearzeit entscheidet. D simuliert m Schritte von T_w, wobei $cm \log m = |w|^2$ gilt. Erklären Sie, wieso T_w nicht notwendigerweise nach Ablauf der m Schritte terminiert.

(c) Sei w so gewählt, sodass T_w eine Turingmaschine ist, die $L(D)$ in Linearzeit entscheidet, deren Simulation durch D aber innerhalb der $|w|^2$ Ausführungsschritte von D nicht terminiert.

Erinnern Sie sich an die Forderung aus Aufgabe 5, dass jede Turingmaschine durch unendlich viele Wörter kodiert sein muss. Zeigen Sie, dass T_w dann auch durch ein anderes Wort w' kodiert wird (d.h., $T_w = T_{w'}$), sodass die Simulation von $T_{w'}$ durch D innerhalb von $|w'|^2$ Schritten terminiert. Folgern Sie, dass $T_w = T_{w'}$ somit $L(D)$ nicht entscheiden kann.