Prof. Martin Hofmann
Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik

Wintersemester 2002/03
31. Oktober 2002

Lösungsvorschläge zur 2. Übung

Komplexitätstheorie

Lösung zu G7:

a) Die Konstruktion geht in etwa so: Für eine Eingabe x 'raten' die Länge $n_1 \in \mathbb{N}$ des potentiellen ersten Wortes x_1 und prüfen ob $x_1 \in L$, was noch Voraussetzung durch Simulation mit einer nicht deterministischen Turing Maschine mglich ist. Endet die Simulation mit Annahme, so raten wir die Länge des nächsten Wortes und simulieren erneut, ansonsten beenden wir sofort mit Ablehnung. Das setzen wir solange fort bis wir die gesammte Eingabe auf diese Weise abgearbeitet haben und akzeptieren, wenn auch die letzte Simulation mit Annahme endete. Offensichtlich gibt es eine akzeptierende Berechnung genau dann, wenn $x \in L^*$ und sonst nicht. Somit erkennt diese nicht deterministische Turing Maschine die Sprache L^*.

   Da nach Annahme $L \in \mathcal{NP}$ jede oben aufgerufene Simulation polynomiell beschränkt durchführbar ist und wir maximal $|x|$ oft auf einer Eingabe der Länge $|x|$ simulieren, ist auch die Laufzeit der oben beschriebenen nicht deterministischen Turing Maschine polynomiell beschränkt durch die Länge der Eingabe x beschränkt, somit folgt $L^* \in \mathcal{NP}$.

b) Wir beschreiben zuerst die zugrundeliegende Idee: Bei Eingabe x mit Länge $n = |x|$ konstruieren wir einen gerichteten Graphen G mit $n+1$ Knoten, so dass die gerichtete Kante $(i, i+j)$ genau dann enthalten ist, wenn an Position i in x ein Wort aus L mit Länge j enthalten ist. Nun suchen wir nach einem gerichtetem Weg von Knoten 1 nach Knoten $n+1$. Existiert dieser, so akzeptieren wir x und verwerfen sonst.

   Die Konstruktion von G benötigt höchstens $O\left(n^2 p(n)\right)$ Schritte, wobei $p(n)$ ein Polynom ist, das die Laufzeit einer deterministischen Turing Maschine beschränkt, welche L erkennt: Für jede Position der Eingabe prüfen wir ab, ob dort ein Wort aus L beginnt, d.h. für n Positionen und jeweils n möglichen Längen ist einmal die Prüfung durchzuführen. Den Graphen G können wir als eine Tabelle kodieren, was bei einem Band ebenfalls quadratische Laufzeit beansprucht. Das Suchen eines Pfades läßt sich ebenfalls polynomiell durchführen, z.B. nach Dijkstra in quadratischer Laufzeit.

   Die Idee der Dynamischen Programmierung spiegelt sich darin wieder, zuerst den Graph zu konstruieren und dann einen Weg zu suchen. Im Gegensatz dazu könnte man auch bei der Pfadsuche jedesmal eine Unterfunktion zur Ermittlung der Teilwörter aufrufen, was jedoch zu einer exponentiellen Laufzeit führen würde.

Lösung zu G8 (3-Dimensionales Matching):

Der Beweis ist etwas umfangreicher und findet sich in ausführlicher Form im Buch "Computers and Intractability" von M.Garey und D.Johnson, ab Seite 50; oder im Vorlesungsskript von D.Karabeg unter `http://www.ifi.uio.no/in210/foils/foils6.pdf` ab Seite 11.

Lösung zu G9: Offensichtlifch gilt $L \in \mathcal{NP}$, denn wir simulieren einfach die i-te nichtdeterministische Turing Maschine auf x und brechen die Simulation nach spätestens t Schritten ab. Zu zeigen bleibt nun noch die polynomielle Reduktion einer beliebigen Sprache $\check{L} \in \mathcal{NP}$ auf L: Wegen $\check{L} \in \mathcal{NP}$ gibt es nichtdeterministische Turing Maschine NT_i, welche Eingaben $x \in \check{L}$ in polynomieller Zeit $p(|x|)$ erkennt. Um nun zu entscheiden, ob ein $x \in \check{L}$ ist oder nicht, müssen wir lediglich prüfen ob $(i, x, \underbrace{0 \cdots 0}_{|x|})) \in L$.

Das zur Reduktion notwendige Kopieren der Eingabe und das Ausrechnen des Polynoms lassen sich in polynomieller Zeit durchführen.

Lösung zu G10: Siehe Beispiellösung zu G6.