AG Theorie der künstlichen Intelligenz
FB Mathematik und Informatik, Universität Bremen
Prof. Dr. Carsten Lutz

MZH 3090
clu@informatik.uni-bremen.de
Tel.: 0421/218-64431

4. Aufgabenblatt für die Vorlesung „Komplexitätstheorie“

Aufgabe 16: 5 Punkte

Das Problem *2-Färbbarkeit (2F)* für ungerichtete Graphen sei analog zu 3-Färbbarkeit definiert, nur mit zwei statt mit drei Farben. Beweise, dass 2F $\in$ P.

Aufgabe 17: 10 Punkte

Konstruiere eine Paddingfunktion für CLIQUE und weise nach, dass die konstruierte Funktion tatsächlich eine Paddingfunktion ist.

Aufgabe 18: 10 Punkte

Viele natürliche Kodierungen von Probleminstanzen als Worte sind nicht surjektiv. Betrachte z.B. SAT und die Kodierung von AL-Formeln über dem Alphabet $\{\neg, \wedge, \vee, 0, 1\}$, wobei Variablen als Worte aus $\{0,1\}^*$ dargestellt werden. Worte wie $0\neg\wedge0110\vee\vee$ sind zwar über diesem Alphabet formuliert aber weder eine “ja”-Instanz noch eine “nein”-Instanz.

Eine Möglichkeit, solche Kodierungen formal zu erfassen, besteht darin, Probleme mit “Versprechen” zu betrachten (promise problems). Sei Σ ein Alphabet. Ein *Problem mit Versprechen* über Σ ist ein Paar $(L_{\text{ja}}, L_{\text{nein}})$ mit $L_{\text{ja}} \subseteq \Sigma^*$, $L_{\text{nein}} \subseteq \Sigma^*$ und $L_{\text{ja}} \cap L_{\text{nein}} = \emptyset$. Es wird nicht gefordert, dass $L_{\text{ja}} \cup L_{\text{nein}} = \Sigma^*$. Intuitiv ist $L_{\text{ja}} \cup L_{\text{nein}}$ das Versprechen bzgl. der Form der Eingabe.

Ein Problem mit Versprechen $(L_{\text{ja}}, L_{\text{nein}})$ ist *in Polyzeit lösbar* wenn es Turingmaschine M und Polynom p gibt, so dass M alle $w \in L_{\text{ja}}$ in Zeit $p(n)$ akzeptiert und alle $w \in L_{\text{nein}}$ in Zeit $p(n)$ verwirft. Für Eingaben $w \in \Sigma^* \setminus (L_{\text{ja}} \cup L_{\text{nein}})$ kann M beliebige Antwort liefern, beliebig viel Zeit brauchen und muss nicht mal anhalten.

Zeige: wenn ein Problem mit Versprechen $(L_{\text{ja}}, L_{\text{nein}})$ in Polyzeit lösbar ist, dann gibt es eine Turingmachine M, die $L_{\text{ja}} \cup L_{\text{nein}}$ löst und auf allen Eingaben $w \in \Sigma^*$ in Polyzeit anhält.

Aufgabe 19: 10 Punkte

Integer Faktorisierung ist das folgende Problem: gegeben $n, m \in \mathbb{N}$ (in binär), entscheide ob m einen Faktor k hat mit $k \leq n$.

Zeige, dass Integer Faktorisierung in NP $\cap$ co-NP ist. Folgere, dass Integer Faktorisierung entweder in P ist oder NP-Intermediate.

Hinweis: Für co-NP verwende Primzerlegung von m und das existierende Resultat, dass von einer binär kodierten Zahl in Polyzeit entschieden werden kann, ob sie prim ist.

Aufgabe 20: 12 Punkte (Zusatzaufgabe)

(a) Es ist bekannt, dass (deterministische) Turingmaschinen ohne Speicher genau die regulären Sprachen REG erkennen. Verwende dieses Resultat, um zu zeigen, dass auch die Turingmaschinen mit konstant grossem Speicher (der zur Verfügung stehende Speicher ist unabhängig von der Eingabe) genau REG erkennen.

(b) Zeige, dass die Sprache

$$L = \{\text{bin}(1)\#\text{bin}(2)\#\cdots\#\text{bin}(k) \mid k \geq 0\}$$

mit Platz $\mathcal{O}(\log(\log(n))$ entschieden werden kann (eine Skizze der TM ist ausreichend). Folgere daraus, dass REG $\subsetneq$ DSpace($\mathcal{O}(\log(\log(n)))$).