Prof. Martin Hofmann
Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik

Wintersemester 2002/03
6. Februar 2003

Lösungsvorschläge zur 12. Übung

Komplexitätstheorie

Lösung zu G40:

a) Sei M eine probabilistische Turing Maschine vom Typ PP für L:

$$
\begin{aligned}
x \in L &\Longleftrightarrow M \text{ akzeptiert } x \text{ mit Wahrscheinlichkeit } > \frac{1}{2} \\
x \notin L &\Longleftrightarrow M \text{ akzeptiert } x \text{ mit Wahrscheinlichkeit } \leq \frac{1}{2} \\
x \in \bar{L} &\Longleftrightarrow M \text{ akzeptiert } x \text{ mit Wahrscheinlichkeit } \leq \frac{1}{2} \\
x \in \bar{L} &\Longleftrightarrow M \text{ verwirft } x \text{ mit Wahrscheinlichkeit } > \frac{1}{2}
\end{aligned}
$$

Somit genügt es, die akzeptierenden und verwerfenden Zustände von M zu vertauschen, um eine probabilistische Turing Maschine zu erhalten, welche unter der PP Interpretation $\bar{L}$ erkennt.

b) Für eine Sprache L sei eine NP Turiing Maschine M gegeben. Es sei M' eine probabilistische Turing Maschine, welche sich zuerst wie M verhält (Wir nehmen o.B.d.A. an, das jeder Berechnungsbaum von M binär und balanciert ist), und am Ende jeder Rechnung, ganz gleich was herauskam, mit Wahrscheinlichkeit $\frac{1}{2}$ akzeptiert und ansonsten das ürsprüngliche Ergebnis liefert.

Somit endet mindestens die Hälfte aller Berechnungen dieser Maschine auf eine Eingabe x mit Akzeptanz; und genau dann, wenn M die Eingabe x akzeptiert (d.h. es gibt mindestens einen Berechnungspfad der zu einem akzeptierenden Zustand führt), akzeptieren mehr als die Hälfte aller Berechnungspfade der neuen probabilistischen Turing Maschine auf Eingabe x.

c) Analog zu vorherigen Aufgabe fügt man noch ein paar Münzwürfe ans Ende der Berechnung an, die ungeachtet der Berechnung zu Akzeptanz oder Ablehnung füren.

Wenn man z.B. eine $\mathrm{PP}_{\frac{1}{4}}$-Maschine hat, fügt man noch zwei Münzwürfe an. Bei zweimal "Zahl" akzeptiert man, ansonsten gibt man das Ergebnis der Berechnung aus, um eine $\mathrm{PP}_{\frac{1}{2}}$-Maschine zu erhalten.

d) Die Klasse BPP bricht die Symmetry zwischen Akzeptanz und Ablehung:

$$x \in L \Longleftarrow M \text{ akzeptiert } x \text{ mit Wahrscheinlichkeit } \geq \frac{1}{2} + \epsilon$$
$$x \notin L \Longleftarrow M \text{ akzeptiert } x \text{ mit Wahrscheinlichkeit } \leq \frac{1}{2} - \epsilon$$

was offensichtlich eine Verstärkung gegnüber PP darstellt.

Lösung zu G41:

a) Da dem Verifier polynomielle Laufzeit erlaubt ist, kann dieser einfach zu beginn die Reduktion durchführen.

b) Eine NP-Maschine kann als “Verifier” interpretiert werden, die jedoch den Prover das “raten” überlässt. Schliesslich kann man die Klasse NP als die Klasse aller Probleme betrachten, deren Lösungen in polynomieller Zeit überprüfbar sind.

Bemerkung: Hier haben wir von der Möglichkeit, den Verifier als probabilistische Turing Maschine zu sehen, noch gar keinen Gebrauch gemacht. In der Tat erhält man genau die Klasse NP, wenn man als Verifier nur nicht-deterministische, polynomiell zeit-beschränkte Turing Maschinen zuläßt.

c) Die Kommunikation zwischen Verifier und Prover ist polynomiell beschränkt, sowohl in der Länge der übertragenen Wörter als auch in der Anzahl der übertragenen Wörter. Somit kann man alle Antwort-Folgen des Provers in Depth-First Order mit nur polynomiellen Platzverbrauch durchprobieren und zählt dabei die endgültigen akzeptierenden Antworten des Verifiers.

d) So wie diese Aufgabe gestellt wurde ist die Antwort leider trivial: Der Prover gibt dem Verifier einfach den passenden Isomorphismus an, den dieser lediglich zu überprüfen hat. Eine Kommunikation vom Verifier zum Prover findet dabei gar nicht statt.

Deutlich interessanter ist hingegen das komplementäre Graphenisomorphieproblem. Der Verifier muss als davon überzeugt werden, das der Prover für zwei gegebene Graphen beweisen kann, das diese *nicht* isomorph sind. Ein herkömmliches IP-System zu diesem Problem wurde in der Vorlesung behandelt. Ein Arthur-Merlin-IP-System dazu ist etwas komplizierter, und wird z.B. in [*U. Schöning,* Perlen der theoretischen Informatik, *BI.-Wissenschaftsverlag, Seite 212*] ausführlich behandelt.

Lösung zu G42:

a) Zu einem gegeben IP-System müssen wir also eine polynomial zeit beschränkte, probabilistische Turing Maschine angeben, welche ein Orakel befragen darf. Den Verfifier kann man ohne Probleme simulieren, jedoch nicht den Prover, da dessen Komplexität nicht weiter eingeschränkt ist - im Gegensatz zur Kommunikation. Wir können daher die Orakelsprache definieren als

$$\left\{ \langle x, y_1, \ldots, y_k, z' \rangle \;\middle|\; P(x, y_1, \ldots, y_k, z) \text{ und } z' \text{ ist Anfangsstück von z} \right\}$$

wobei $P(x, y_1, \ldots, y_k, z)$ bedeutet das der Prover bei Eingabe x mit z antwortet, wenn der Verfier zuvor schon die k-vielen Anfragen $y_1, \ldots, y_k$ gestellt hat, den schließlich kann sich der Prover auf seinem Arbeitsband ja beliebige Informationen merken. Dem simulierten Verifier ist es jedoch aufgrund der polynomiellen Beschränkung der Kommunkation möglich, sich seine vorhergenden Anfragen zu merken und somit durch Orakelfragen die Antwort des Provers zu erhalten.

b) Bei der Umkehrung besteht das Problem genau darin, dass sich der Prover vorhergende Anfragen merken kann und so z.B. bei einer identischen Anfrage eine andere Antwort geben kann. In dem Falle eine gültigen Anfrage ergibt sich daraus kein Problem, da man ja einem optimalen Prover wählen kann, der gemäß dem Orakel antwortet. Falls die Eingabe jedoch verworfen werden muß, so darf sich der Verifier von *keinem* Prover täuschen lassen. Es ist nicht klar, wie dies gesichert werden kann.