RWTH Aachen **WS 2012/13**
Lehrgebiet Theoretische Informatik Probeklausur
Reidl–Ries–Rossmanith–Sanchez–Tönnis 25.01.2013

Probeklausur zur Vorlesung Berechenbarkeit und Komplexität

Aufgabe 1 (1+2+6+3 Punkte)

Sind die folgenden Sprachen rekursiv oder nicht? Beweisen Sie Ihre Aussage!

a) $L_a = \emptyset$

b) $L_b = \{ \langle M \rangle \mid L(M) = \emptyset \}$

c) $L_c = \{ \langle M_1 \rangle \langle M_2 \rangle \mid \langle M_2 \rangle \in L(M_1) \}$

d) $L_d = \{ \langle M \rangle \mid M \text{ hält auf } \epsilon \text{ in } \langle M \rangle \text{ Schritten} \}$

Lösungsvorschlag.

Im Folgenden ist Anzunehmen, daß Turingmaschinen, welche Gödelnummern als Eingabe erwarten, alle anderen Eingaben sofort verwerfen.

a) Ist natürlich entscheidbar durch eine Turingmaschine, die jede Eingabe verwirft. Auch ist L_a schon deshalb entscheidbar, weil sie eine endliche Sprache ist.

Bepunktung: Um den einzigen Punkt aus dieser Aufgabe zu kriegen, muß man diesen einfachen Fakt erwähnen, keine besondere Begründung ist nötig.

b) Diese Sprache ist nicht rekursiv. Die einfachste Art und Weiße das zu beweisen ist mit dem Satz von Rice: definiere

$$S = \{f \in \mathcal{R} \mid \ f(w) \in \{0, \perp\} \text{ für alle } w \in \{0,1\}^*\}$$

Dann ist $L_b = \{\langle M \rangle \mid f_M \in S\}$, denn L_b enthält gerade alle Gödelnummern von Turingmaschinen, die alle Wörter nicht akzeptieren. Es bleibt zu zeigen, daß S weder vollständig noch leer ist. Beides ist einfach ersichtlich: es gibt eindeutig Turingmaschinen, die kein Wort akzeptieren (etwa diejenge, die jede Eingabe verwirft) und solche, die mindestens ein Wort akzeptieren. Durch Anwendung des Satzes erhalten wir, daß L_b nicht entscheidbar ist.

Bepunktung: Hier gibt es einen Punkt für die Menge S und einer Begründung dafür, daß diese Menge von turingberechenbaren Funktionen die angegeben Sprache modelliert. Den zweiten Punkt gibt es für die begründete Aussage, daß S weder ganz $\mathcal{R}$ ist, noch die leer ist. Dafür sollte man triviale Funktionen finden die nicht in bzw. in S sind. Daß diese triviale Funktionen turingberechenbar sind, muß nicht weiter ausgeführt werden.

c) Wir zeigen per Unterprogrammtechnik, daß L_c nicht entscheidbar ist. Nehmen wir an, es existiere eine TM M_c die L_c entscheiden. Wir können dann zeigen, daß wir eine Turingmaschine M_ϵ konstruieren können, die das spezielle Halteproblem H_ϵ entscheidet: Als erstes verwirft M_ϵ die Eingabe $\langle M \rangle$, wenn diese keine gültige Gödelnummer ist. Wenn $\langle M \rangle$ gültig ist, dann konstruiert M_ϵ aus $\langle M \rangle$ das Wort $\langle M^* \rangle \langle M \rangle$, wobei M^* eine feste Turingmaschine ist, welche ihre Eingabe als Gödelnummer interpretiert, die darin kodierte Turingmaschine auf der leeren Eingabe simuliert und anschließend akzeptiert—d.h. $L(M^*)$ enthält genau diejenigen Gödelnummern, die Turingmaschinen kodieren welche auf der leeren Eingabe halten.

Nun nutzt M_ϵ die hypothetische Maschine M_c als Unterprogramm mit Eingabe $\langle M^* \rangle \langle M \rangle$ und übernimmt die Antwort von M_c. Beweisen wir, daß M_ϵ tatsächlich H_ϵ entscheidet:

$$\begin{aligned} \langle M \rangle \in H_\epsilon &\Leftrightarrow M \text{ hält auf } \epsilon \\ &\Leftrightarrow \langle M \rangle \in L(M^*) \\ &\Leftrightarrow \langle M^* \rangle \langle M \rangle \in L_c \\ &\Leftrightarrow M_c \text{ akzeptiert } \langle M^* \rangle \langle M \rangle \\ &\Leftrightarrow M_\epsilon \text{ akzeptiert } \langle M \rangle \end{aligned}$$

Die TM M^* existiert eindeutig, da diese nur eine einfache Abwandlug der universellen TM ist. Da diese Konstruktion das spezielle Halteproblem lösen würde, kann M_c nicht existieren. Somit ist bewiesen, daß L_c nicht rekursiv ist. □

Bepunktung: Zu erkennen, daß die Sprache nicht rekursiv ist, gibt bereits einen Punkt. Zwei Punkte gibt es für die Konstruktion von M_ϵ. Hier hilft es, eine Skizze zu machen. Eine Erklärung der Konstruktion in Textform ist trotzdem nötig. Einen weiteren Punkt gibt es für das Erwähnen der Konstruierbarkeit der einzelnen Schritte. Die letzten zwei Punkte werden für den Beweis, daß M_ϵ das spezielle Halteproblem löst, vergeben.

d) L_d ist berechenbar: wir konstruieren eine Turingmaschine M_d, welche ihre Eingabe als Gödelnummer interpretiert und die darin kodierte Turingmaschine M für $\langle M \rangle$ Schritte auf der leeren Eingabe simuliert. Hält M in dieser Zeit, so akzeptiert M_d, ansonsten verwirft sie. Da M_d maximal $O(\langle M \rangle)$ Schritte läuft, hält sie auf jeder Eingabe und entscheidet somit die Sprache L_d.

Bepunktung: Einen Punkt gibt es für die Aussage, daß die Sprache rekursiv ist. Einen weiteren Punkt gibt es für die Angabe einer TM, welche die Sprache entscheidet. Der letzte Punkt wird für den Beweis, daß die angegebene TM auf jeder Eingabe hält, gegeben. Wie man an der angegebenen Musterlösung sehen kann, muß die TM die, welche Sprache entscheidet, nicht besonders genau beschrieben werden, da diese ja sehr einfach ist.

Aufgabe 2 (10 Punkte)

Zeigen Sie, daß die universelle Sprache $L_U = \{ \langle M \rangle w \mid w \in L(M) \}$ rekursiv aufzählbar ist, indem Sie einen Aufzähler M_{L_U} angeben.

Zur Erinnerung: Ein Aufzähler ist eine Turing-Maschine mit einem zusätzlichen endlosen Ausgabeband, auf die sie alle Wörter, die in der aufzuzählenden Sprache enthalten sind, in beliebiger Reihenfolge (evtl. mehrfach) schreibt. Das Ausgabeband kann von der Maschine *nicht* gelesen werden.

Lösungsvorschlag.

Zunächst stellen wir fest, daß die Wörter der Sprache L_U durch eine Turingmaschine erkannt werden können: für ein Wort $\langle M \rangle w \in L_U$ simuliert man schlicht M auf w und akzeptiert, wenn M hält.

Damit ist L_U semi-entscheidbar durch eine Turingmaschine $M_{L_U}^{semi}$ und wir wissen aus der Vorlesung, daß es daher auch einen Aufzähler für L_U geben muß. Wir folgen der kanonischen Konstruktion eines solchen Aufzählers und erhalten M_{L_U} wie folgt: wir schreiben sukzessive alle Wörter der Sprache $\{0,1\}^*\Sigma^*$ in der kanonischen Reihenfolge auf das Arbeitsband. Nach dem Hinzufügen des i-ten Wortes w_i simulieren wir $M_{L_U}^{semi}$ auf den Wörtern $w_1, \ldots, w_i$ für i Schritte—akzeptiert $M_{L_U}^{semi}$ eines dieser Wörter, so geben wir es aus.

Da $L_U \subseteq \{0,1\}^*\Sigma^*$, ist M_{L_U} ein Aufzähler für L_U.

Bepunktung: Drei Punkte gibt es schon dafür, zu erkennen, daß die angegeben Sprache erkannt werden kann. Für die Beschreibung eines Aufzählers werden die restlichen sieben Punkt verteilt. Für jede Ungenauigkeit oder Lücke in der Beschreibung gäbe es einen Punktabzug. Benutzt man die Konstruktion aus der Vorlesung, ist kein Beweis der Korrektheit nötig.

Wenn man eine Konstruktion für den Aufzähler gibt, die nicht diese der Vorlesung ist (was natürlich auch möglich ist), müsste man dann doch einen Korrektheitsbeweis angeben. Volle Punktzahl gibt es dann für einen korrekten und lückenlosen Beweis. Punkte werden für jeden Fehler abgezogen. Es lohnt sich also, die Ergebnisse aus der Vorlesung zu lernen!

Aufgabe 3 (6 Punkte)

Ein Evolutionsbiologe steht vor folgender Aufgabe: Für eine Menge von nah verwandten Spezies soll anhand eines bestimmten DNA-Abschnitts entschieden werden, wie dieser Abschnitt wohl für den gemeinsamen evolutionären Vorfahren aussah. Als Arbeitshypothese soll dazu eine DNA-Sequenz berechnet werden, die allen gegebenen Abschnitten möglichst ähnlich sein soll (alle DNA-Abschnitte besitzen die gleiche Länge n).

Als Maß entscheidet sich der Biologe für die maximale *Hammingdistanz* der berechneten Sequenz zu allen gegebenen; die Distanz der berechneten Sequenz sollte also zu jeder anderen Sequenz höchstens t betragen.

Zeigen Sie, daß dieses Problem in NP liegt.

Hinweis: Die Hammingdistanz zweier Sequenzen ist definiert als die Anzahl der Positionen, an denen sie sich unterscheiden.

Lösungsvorschlag.

Dieses Problem hat ein einfaches Zertifikat, nämlich die Sequenz w, welche zu allen Eingabesequenzen $w_1, \ldots, w_n$ höchstens Hammingdistanz t hat. Das Zertifikat kann einfach in polynomieller Zeit überprüft werden, dazu berechnen wir für jedes $1 \leq i \leq n$ die Hammingdistanz zwischen w und w_i—ist die Distanz größer als t für irgendein i, so verwerfen wir. Das berechnen der Hammingdistanz selbst ist in $O(n)$ Schritten einfach machbar:

wir vergleichen jede Stelle der zwei Wörter und zählen diejenigen Stellen, an denen sie sich unterscheiden.

Bepunktung: Drei Punkte gibt es schon für die Idee, daß eine Lösungssequenz w bereits ein Zertifikat des Problems darstellt. Die anderen drei Punkte gibt es für eine Erläuterung dafür, wie dieses Zertifikat in polynomieller Zeit überprüft werden kann. Die Angabe, daß dieses sogar linearer Zeit funktioniert, ist nicht nötig.

Vor allem ist es wichtig, in so einer Aufgaben die Aufgabenstellung genau zu lesen und keine Zeit damit zu verlieren, die NP-schwere des Problems zu Beweisen.

Aufgabe 4 (12 Punkte)

Bei der Versteigerung von Funkfrequenzen für Mobilfunknetze ist der Preis, den ein Bieter bereit zu zahlen ist, von der Anzahl der ihm zugesprochenen Frequenzbänder abhängig—welche Frequenzen er erhält, spielt keine Rolle.

Formalisiert wird diese Situation durch das Problem Multi-Unit Auction: Eine große Anzahl m gleichartiger Gegenstände (hier Frequenzbänder) wird an n Bieter versteigert, wobei jeder Bieter i einen unterschiedlichen Preis für die Anzahl s_i der im zugesprochenen Gegenstände zahlt. Dieser Preis wird für diesen Bieter als Valuierungsfunktion $v_i\colon \{1,\dots,m\} \to \mathbb{Q}$ modeliert.

Beim Entscheidungsproblem Multi-Unit Auction ist gefragt, ob eine Verteilung $s = (s_1, s_2, \dots, s_n)$ der Gegenstände existiert, so daß der Gesamtgewinn $\sum_{1\le i\le n} v_i(s_i)$ mindestens T beträgt.

Zeigen Sie durch die Reduktion Knapsack $\le_p$ Multi-Unit Auction, daß dieses Problem NP-schwer ist.

Multi-Unit Auction	
Eingabe:	Eine Zahl $m \in \mathbb{N}$, n monoton steigende Valuierungsfunktionen $v_i\colon \{1,\dots,m\} \to \mathbb{Q}$ und eine Zahl $T \in \mathbb{N}$
Problem:	Existiert ein Tupel $s = (s_1, s_2, \dots, s_n)$ mit $\sum_{1\le i\le n} s_i \le m$, so daß $\sum_{1\le i\le n} v_i(s_i) \ge T$?

Knapsack	
Eingabe:	Eine Menge von Objekten I, wobei jedem Objekt $i \in I$ die Größe $w_i \in \mathbb{N}$ und der Wert $v_i \in \mathbb{N}$ zugeordnet ist. Außerdem ein Maximalgewicht $w \in \mathbb{N}$ und ein Minimalwert $v \in \mathbb{N}$.
Problem:	Existiert eine Teilmenge $I' \subseteq I$ der Objekte, für die gilt, daß $\sum_{i\in I'} w_i \le w$ und $\sum_{i\in I'} v_i \ge v$?

Hinweis: Nehmen Sie an, daß die Valuierungsfunktionen einer Multi-Unit Auction-Instanz als Wertetabellen kodiert werden.

Lösungsvorschlag.

"⇒" Gegeben sei eine KNAPSACK-Instanz $\mathcal{I}_{Kn}$ mit n Objekten I mit Gewichten $w_1, \ldots, w_n$ und Werten $v_1, \ldots, v_n$ sowie Maximalgewicht w und Mindestwert v. Daraus konstruieren wir eine MULTI-UNIT AUCTION-Instanz $\mathcal{I}_{MUA}$ wie folgt:

$$
\begin{aligned}
m &:= w \\
v_1(s) &:= \begin{cases} 0 \text{ für } s < w_1 \\ v_1 \text{ sonst} \end{cases} \\
&\vdots \\
v_n(s) &:= \begin{cases} 0 \text{ für } s < w_n \\ v_n \text{ sonst} \end{cases} \\
T &:= v
\end{aligned}
$$

Dies ist schon eine vollständige Reduktion. Die darin benutzten Valuierungsfunktionen können mit sehr wenige Platz kodiert werden.

(Hier ist zu beachten, daß man nicht eine komplette Wertetabelle kodieren würde, da m Einträge bereits exponentiell viele in der Kodierungslänge von m wären. Stattdessen kodiert man die Funktionen v_i, indem man nur die Stellen x mit $v_i(x) \neq v_i(x-1)$ angibt. Da die Aufgabenstellung an dieser Stelle unklar war, wird dieser Aspekt in der Bepunktung nicht berücksichtigt.)

Es fehlt noch zu zeigen, daß diese Reduktion korrekt ist.

Zeigen wir zuerst, daß wenn die KNAPSACK eine Lösung hat, dann hat die MULTI-UNIT AUCTION Instanz die wir daraus konstruieren auch eine Lösung. Wir nehmen also an, daß $\mathcal{I}_{Kn} \in$ KNAPSACK, d.h. es existiert eine Menge $I' \subseteq I$ der Objekte, so daß $\sum_{i \in I'} w_i \leq w$ und $\sum_{i \in I'} v_i \geq v$. Dann ist das Tupel $s = (s_1, \ldots, s_n)$ definiert durch

$$
s_i := \begin{cases} w_i \text{ falls } i \in I' \\ 0 \text{ sonst} \end{cases}
$$

eine Lösung der Instanz $\mathcal{I}_{MUA}$, denn es gilt

$$
\begin{aligned}
\sum_{1 \leq i \leq n} s_i &= \sum_{i \in I'} w_i \leq w = m \\
\sum_{1 \leq i \leq n} v_i(s_i) &= \sum_{i \in I'} v_i(w_i) = \sum_{i \in I'} v_i \geq v = T
\end{aligned}
$$

und somit ist $\mathcal{I}_{MUA} \in$ MULTI-UNIT AUCTION.

"⇐" Es fehlt noch zu zeigen, daß wenn unsere MULTI-UNIT AUCTION-Instanz eine Lösung hat, dann hat unsere ursprüngliche KNAPSACK-Instanz auch eine Lösung.

Sei das Tupel $s = (s_1, \ldots, s_n)$ eine Lösung der Instanz $\mathcal{I}_{MUA}$. Sei $I' = \{i \mid s_i \geq w_i\}$. Dann gilt

$$\sum_{i \in I'} w_i = \sum_{1 \leq i \leq n} s_i \leq m = w$$
$$\sum_{i \in I'} v_i = \sum_{i \in I'} v_i(w_i) = \sum_{1 \leq i \leq n} v_i(s_i) \geq T = v$$

Somit ist bewiesen, daß es dann eine Lösung der Originalinstanz gibt. □

Bepunktung: Für die korrekte Konstruktion gibt es drei Punkte. Für den Beweis, daß man eine Lösung von KNAPSACK in eine Lösung für MULTI-UNIT AUCTION umwandeln kann, gibt es vier Punkte. Für den Beweis, daß man Lösungen von der MULTI-UNIT AUCTION in Lösungen für die ursprüngliche KNAPSACK-Instanz zurückwandeln kann, gibt es fünf Punkte.

Um die volle Punktzahl zu erreichen, muß der Beweis mindestens so vollständig sein wie in dieser Musterlösung. Man könnte natürlich eine andere Konstruktion angeben, der Beweis jedoch muß aus diesen drei Teilen bestehen. Für jede Ungenauigkeit im Beweis gibt es dann einen Punktabzug. Bei Reduktionen erwarten wir eine präzise Beschreibung. Eine mathematische Formulierung der Reduktion—ähnlich der obigen—hilft dabei.