RWTH Aachen | **WS 2003/04**
Lehrgebiet Theoretische Informatik | Lösung zur Probeklausur
Mölle—Richter—Rossmanith | 09.01.04

Übung zur Vorlesung Berechenbarkeit und Komplexität

Aufgabe 1 (1 Punkt)

Die drei gängigsten Varianten sind Adjazenzmatrizen, Adjazenzlisten und Inzidenzlisten. Bei den ersten beiden wird in einer 0-1-Matrix bzw. in knotenbezogenen Listen festgehalten, welche Knoten mit welchen anderen Knoten durch Kanten verbunden sind. Bei der dritten Variante hingegen wird für jeden Knoten eine Liste der mit ihm verknüpften Kanten verwendet. Offenbar lassen sich alle derartige Repräsentationen in Bitstrings kodieren.

Punkte: Bei einer korrekten Nennung wird ein halber, bei mehr korrekten Nennungen ein ganzer Punkt vergeben.

Aufgabe 2 (3 Punkte)

1. Für alle Sprachen L gilt $\lambda \in (\overline{L^*})^*$, da der Kleensche Abschluss stets das leere Wort enthält.

2. Es kann keine Sprache L mit der Eigenschaft $\lambda \in \overline{L} \cdot L$ geben, da λ in genau einer der beiden Sprachen L und $\overline{L}$ enthalten ist. Jedes Wort aus $\overline{L} \cdot L$ hat mindestens die Länge eins.

3. Für L mit $\lambda \notin L$ gilt, dass $\lambda \notin h(L)$ ist, weil h injektiv ist (für jedes $w \neq \lambda$ mit $h(w) = \lambda$ gälte $h(ww) = h(w)$, womit h nicht injektiv wäre). Für L mit $\lambda \in L$ muss auch $\lambda \in h(L)$ gelten, da nach Definition $h(\lambda) = \lambda$ ist.

Punkte: Pro richtiger Antwort mit Begründung gibt es einen Punkt.

Aufgabe 3 (2 Punkte)

1. Für die Kodierung von k genügen $\log k$ (genau: $\lceil \log (k+1) \rceil$) viele Bits. Also ist $K(n) \leq \log k + O(1) = \log \sqrt[2^{100!}]{n} + O(1)$.

2. Die Diagonale einer $k \times k$-Matrix hat k Elemente, es gibt also genau 2^k verschiedene 0-1-Diagonalmatrizen dieser Größe. Somit gilt $K(n) \leq \log k + O(1) = \log \log n + O(1)$.

Punkte: Pro richtiger Teilantwort (Komplexität in k bzw. n) wird ein halber Punkt vergeben.

Aufgabe 4 (3+1 Punkte)
Die Turingmaschine durchläuft die Eingabe aus $\{a, b\}^*$, hängt drei #-Zeichen an und läuft dann wieder an den Anfang der Eingabe. Anschließend wird in der Eingabe nach Vorkommen des Zeichens a gesucht. Sobald ein a gefunden wird, löscht die Maschine dieses a und im Anschluss ein # vom Band. Genau dann, wenn die Eingabe ursprünglich dreimal das Zeichen a enthielt, werden auf diese Weise alle Vorkommen von a und # entfernt. In diesem Fall liest die Maschine nur noch b und $\sqcup$, woraufhin akzeptiert wird. Andernfalls wird entweder ein viertes a gelesen (und verworfen) oder am Ende noch ein # gelesen (und verworfen).
Die gesuchte Sprache ist also $L = \{w \in \{a, b\}^* \mid |w|_a = 3\}$.
Es ist L regulär und ein passender endlicher Automat schnell konstruiert: In Zuständen q_0 bis q_4 wird die Anzahl der bisher gelesenen as kodiert (0, 1, 2, 3, 4 oder mehr). Lediglich der Zustand q_3 ist auch ein Endzustand. Die einzigen Transitionen, die keine Schleifen sind, sind $\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_2$, $\delta(q_2, a) = q_3$ und $\delta(q_3, a) = q_4$.
Punkte: Für die korrekte Benennung der gesuchten Sprache gibt es zwei Punkte, bei näherer Beschreibung der Turingmaschine einen weiteren Punkt und für die Angabe eines korrekten Automaten ebenfalls einen Punkt.

Aufgabe 5 (5 Punkte)

1. Nach dem Satz von Rice kann die Sprache $\{$ Kod(M) | „Paulson-Kreis“ $\in L(M)$ $\}$ nicht rekursiv sein, da sie ein nichttriviales semantisches Entscheidungsproblem darstellt: Es gibt Turingmaschinen, die das genannte Wort akzeptieren, und ebenso solche, die es verwerfen. Die dritte Bedingung wird dadurch erfüllt, dass zwei Turingmaschinen, die die gleiche Sprache akzeptieren, das genannte Wort stets übereinstimmend akzeptieren oder nicht akzeptieren.

2. Der Satz von Rice ist hier nicht anwendbar. Es gibt beispielsweise zwei Turingmaschinen M_1 und M_2 mit $L(M_1) = L(M_2) = \emptyset$, von denen eine das Wort „Paulson-Kreis“ verwirft, während die andere darauf gar nicht hält. Man kann aber reduzieren:
   Gegeben sei eine Turingmaschine M_H, für die das λ-Halteproblem vorliegt (es wird gefragt, ob M_H auf λ hält). Natürlich können wir M_H so modifizieren, dass die Eingabe λ anfangs durch „Paulson-Kreis“ ersetzt und die Zustände q_{accept} und q_{reject} vertauscht werden. Wenn also die Sprache $\{$ Kod(M) | M verwirft das Wort „Paulson-Kreis“ $\}$ rekursiv wäre, dann auch $L_{H,\lambda}$. Widerspruch.

   Etwas einfacher ist es, Folgendes einzusehen: Wäre diese Sprache rekursiv, dann könnte man mit ihrer Hilfe die aus der ersten Teilaufgabe entscheiden, indem man einfach die Zustände q_{accept} und q_{reject} in der Eingabeturingmaschine vertauscht.

3. Die Sprache $\{$ Kod(M) | $L(M)$ ist nicht rekursiv aufzählbar $\}$ ist leer und somit rekursiv, weil jede Turingmaschine M genau die Wörter aus $L(M)$ akzeptiert. Deshalb ist $L(M)$ immer rekursiv aufzählbar.

4. Für eine Turingmaschine M mit $L(M) = L(M)^{24}$ gilt offenbar $L(M) = \{\lambda\}$. Somit ist der Satz von Rice anwendbar, die hier betrachtete Sprache also nicht rekursiv.

Punkte: Pro richtiger Antwort mit Begründung gibt es einen Punkt, bei der zweiten Teilaufgabe sogar zwei Punkte.

Aufgabe 6 (-7 bis 7 Punkte)

1. Das Halteproblem L_H ist nicht rekursiv, aber rekursiv aufzählbar. Sein Komplement ist somit nicht rekursiv aufzählbar.
2. Das Postsche Korrespondenzproblem PKP ist nicht rekursiv, aber rekursiv aufzählbar. Sein Komplement ist somit nicht rekursiv aufzählbar.
3. Das Modifizierte Postsche Korrespondenzproblem MPKP ist nicht rekursiv, aber rekursiv aufzählbar. Sein Komplement ist somit nicht rekursiv aufzählbar.
4. Die Diagonalsprache L_{diag} ist nicht rekursiv aufzählbar. Ihr Komplement ist nicht rekursiv, aber rekursiv aufzählbar.
5. Die universelle Sprache L_U ist nicht rekursiv, aber rekursiv aufzählbar. Ihr Komplement ist somit nicht rekursiv aufzählbar.
6. Die Sprache der Ordnungszahlen von Turingmaschinen, die auf allen Eingaben halten, ist nicht rekursiv aufzählbar. Ihr Komplement, also die Sprache der Ordnungszahlen von Turingmaschinen, die auf mindestens einer Eingabe nicht halten, ist nicht rekursiv aufzählbar.

Punkte: Pro Teilaufgabe gibt es für die korrekte Beurteilung der Sprache und ihres Komplements einen Punkt, für eine nicht völlig richtige Beurteilung einen Minuspunkt. Unbeantwortete Teilaufgaben ergeben null Punkte. Einen zusätzlichen Minuspunkt erhält, wer für eine Sprache L behauptet, L sei nicht rekursiv, aber rekursiv aufzählbar, und $\overline{L}$ rekursiv aufzählbar. Einen zusätzlichen Pluspunkt gibt es für die komplett korrekte Beantwortung der letzten Teilaufgabe (6).

Aufgabe 7 (2 Punkte)

Diese Aufgabe ist auf der Probeklausur fälschlicherweise als Aufgabe 8 bezeichnet.

Man kann hier ausschließlich mit der ersten Karte beginnen, da nur hier die beiden enthaltenen Wörter ein gemeinsames Präfix haben (nämlich 1). Danach ist die Anzahl der Nullen, die in den beiden entstehenden Strings insgesamt vorkommen, ungerade (anfangs eins). Da alle anderen Karten jeweils eine gerade Anzahl von Nullen enthalten, bleibt die Gesamtzahl der Nullen immer ungerade, die beiden entstehenden Zeichenketten können also niemals gleich werden.

Punkte: Für einen korrekten Beweis gibt es zwei Punkte. Bei leichten Fehlern, aber funktionierendem Ansatz wird ein Punkt vergeben.

Aufgabe 8 (2 Punkte)

Eine naheliegende Partitionierung von $\mathbf{R}$ in Mengen gleicher Größe ergibt sich durch die Zerlegung in Intervalle:

$$\ldots, M_3 = [-2, -1[, M_1 = [-1, 0[, M_2 = [0, 1[, M_4 = [1, 2[, M_6 = [2, 3[, \ldots$$

Offenbar sind je zwei dieser Mengen gleich mächtig, da sich einfach eine Bijektion der Art $f(i) = i + k$ angeben lässt, die Elemente von der einen in die andere Menge überführt.

Die Mengen sind jeweils reelle Intervalle, also überabzählbar und damit genauso groß wie die Originalmenge.

Punkte: Für eine korrekte Partitionierung mit Nachweis der Gleichmächtigkeit gibt es zwei Punkte, bei kleineren Fehlern wird davon ein Punkt abgezogen.

Aufgabe 9 (4 Punkte)

Es gibt abzählbar unendlich viele Turingmaschinen, also auch höchstens abzählbar viele berechenbare Funktionen. Die Klasse der Funktionen von $\mathbf{N}$ nach $\{0,1\}$ ist bereits überabzählbar; ihre Mächtigkeit beträgt $|\mathcal{P}(\mathbf{N})|$ (Satz von Cantor anwenden). Daraus folgt, dass die Klasse der Funktionen von $\mathbf{N}$ nach $\mathbf{N}$ ebenfalls überabzählbar ist. Also sind die meisten dieser Funktionen nicht berechenbar.

Der Ausdruck „die meisten" bedeutet hier, dass es unter den überabzählbar vielen Elementen der betrachteten Klasse lediglich abzählbar unendlich viele Ausnahmen gibt. Anschaulich wird dies durch den folgenden Fakt: Wenn wir zufällig gemäß Gleichverteilung eine Funktion von $\mathbf{N}$ nach $\mathbf{N}$ wählen, dann ist sie mit Wahrscheinlichkeit 0 berechenbar.

Punkte: Für die Erwähnung der Abzählbarkeit der Turingmaschinen wird ein Punkt, für den Nachweis der Überabzählbarkeit der untersuchten Funktionenmenge werden zwei Punkte vergeben. Alternative Beweise, sofern korrekt, werden ebenfalls mit bis zu drei Punkten belohnt. Einen weiteren Punkt gibt es für eine Erklärung des Begriffs „die meisten", die inhaltlich der obigen Erörterung gleichwertig ist. Dabei war die anschauliche Erklärung mittels Wahrscheinlichkeiten nicht verlangt.

Aufgabe 10 (3+4 Punkte)

1. Die Klasse der Funktionen von $\mathbf{N}$ nach $\{0,1\}$ ist bereits überabzählbar; ihre Mächtigkeit beträgt $|\mathcal{P}(\mathbf{N})|$. Nach dem Satz von Cantor ist diese Menge mächtiger als $\mathbf{N}$, also überabzählbar. Eine vollständige Nummerierung dieser Funktionen mit natürlichen Zahlen kann es also nicht geben.

2. Die Menge aller Permutationen über einer Menge M ist gerade die Menge aller Bijektionen von M nach M. Man sieht leicht, dass jede Funktion f von $\mathbf{N}$ nach $\{0,1\}$ auch als Bijektion von $\mathbf{N}$ nach $\mathbf{N}$ kodiert werden kann, indem Funktionswerte 0 durch gerade und Funktionswerte 1 durch ungerade natürliche Zahlen ersetzt werden. Da diese Kodierung injektiv und die Menge der Funktionen von $\mathbf{N}$ nach $\{0,1\}$ überabzählbar ist, folgt für abzählbar unendliche Mengen M, dass die Menge aller Permutationen über M überabzählbar ist.

   Es gibt abzählbar unendlich viele Turingmaschinen und somit überabzählbar viele Anordnungen. Da die Tante aber lediglich unendlich abzählbar viele Minuten zur Verfügung hat, wird sie nicht alle Anordnungen durchtesten können.

Punkte: Für korrekte Beweise gibt es drei bzw. vier Punkte. Wenn der Ansatz stimmt, aber kleine Fehler auftauchen, gibt es natürlich Punktabzug.

Aufgabe 11 (3+5 Punkte)

1. Ja. Man kann die Maschine M nämlich so modifizieren, dass sie bei Start zum Zeitpunkt t zuerst in die Zukunft springt ($t+3$ Sek.), dort völlig normal rechnet und

beim Akzeptieren stets zum Zeitpunkt $t+1$ Sek. zurückspringt. Genau dann, wenn die Eingabe w in $L(M)$ liegt, wird also innerhalb einer Sekunde akzeptiert. Wenn andernfalls zwei Sekunden verstreichen, ohne dass M akzeptiert, kann verworfen werden.

2. Um einen Effekt wie oben zu erreichen, kann eine ZRTM etwa wie folgt konstruiert werden: Es gibt ein Extraband, auf das eine Zahl geschrieben werden kann. Wenn in den speziell ausgezeichneten Sprungzustand q_1 gegangen wird, soll die Zahl auf diesem Band als Zeitpunkt interpretiert und dahin gesprungen werden. Es gibt dann abzählbar unendlich viele ZRTMs. Dies gilt u.E. für *alle sinnvollen* Auffassungen von ZRTMs.

   Ein einfaches (nicht-konstruktives) Argument für die Existenz nicht-ZRTM-entscheidbarer Sprachen ist, dass es überabzählbar viele Sprachen über dem Alphabet $\{0,1\}$ gibt.

   Zum konstruktiven Nachweis kann hier eine an Zeitreisen-TMs angepasste Diagonalsprache verwendet werden.

   Wenn die Menge aller Zeitreisen-Turingmaschinen abzählbar ist, können diese Maschinen nummeriert werden. Für die Menge der Wörter aus $\{0,1\}^*$ gilt dies allemal. Wie gehabt sei L_{diag} die Menge aller Wörter w_i, so dass M_i das Wort w_i nicht akzeptiert.

   Es sei nun M eine ZRTM mit $L(M) = L_{\text{diag}}$. Natürlich gibt es ein $j \in \mathbf{N}$ mit $M = M_j$. Da $w_j \in L_{\text{diag}} = L(M_j)$ gilt, muss M_j auch das Wort w_j akzeptieren. Andererseits gilt aber für jedes $w_i \in L_{\text{diag}}$ die Aussage $w_i \notin L(M_i)$, also insbesondere $w_j \notin L(M_j)$. Widerspruch.

Punkte: Für korrekte Beweise gibt es drei bzw. fünf Punkte. Wenn der Ansatz stimmt, aber kleine Fehler auftauchen, gibt es natürlich Punktabzug. Bei der zweiten Teilaufgabe gibt es sicher viele Möglichkeiten zur Lösung, wichtig ist nur, dass es sich um einen korrekten Beweis handelt.

Punkteverteilung

-7 bis 21	nicht bestanden
22 bis 27	ausreichend
28 bis 33	befriedigend
34 bis 39	gut
40 bis 45	sehr gut