Theoretische Informatik 2 – Übungsblatt 11 mit Musterlösungen

(LV-Nr.: 185.183, 185.231, WS 2004/2005)

Thema: Komplexität, Nichtdeterminismus

Übung 11.1 *Sei $P\colon(1, A)$ das folgende Programm zur nichtdeterministischen[1] Turingmaschine* NTM_1^1 *mit*

$$
\begin{aligned}
A = \{1\colon\ & R\ \underline{\text{then}}\ 2,\\
2\colon\ & \underline{\text{if}}\ t\ \underline{\text{then}}\ 4\ \underline{\text{else}}\ 3,\\
3\colon\ & R\ \underline{\text{then}}\ 2,\\
4\colon\ & d_I\ \underline{\text{then}}\ 3,\\
4\colon\ & L\ \underline{\text{then}}\ 5,\\
5\colon\ & \underline{\text{if}}\ t\ \underline{\text{then}}\ 0\ \underline{\text{else}}\ 6,\\
6\colon\ & L\ \underline{\text{then}}\ 5 \quad \}.
\end{aligned}
$$

Welche Relation berechnet das NTM_1^1*-Programm P, d.h. welches sind die möglichen Endergebnisse bei Eingabe $n \in \mathbb{N}$ (wobei Ein- aus Ausgaben unär kodiert sind)?*

Lösung: Die Arbeitsweise des Programms ist wie folgt: Zunächst wird die Eingabe n vollständig gelesen, bis der Lese-Schreibkopf über dem ersten "B" (Blank) rechts der Eingabe steht. Sodann wird nichtdeterministisch verzweigt: Entweder wird (siehe erste Anweisung mit Marke 4) ein "I" angehängt und die (verlängerte) Eingabe weiter gelesen, oder (siehe zweite Anweisung mit Marke 4) die Berechnung beendet, indem an den Ausgangspunkt zurückgegangen wird. Offensichtlich ist daher die Menge der möglichen Endergebnisse bei Eingabe n

$$\{k \in \mathbb{N} \mid k \geq n\}\ .$$

Übung 11.2 *Geben Sie ein Programm $P\colon(1, A)$ zu einer nichtdeterministischen Turingmaschine* NTM_1^1 *an, das bei Eingabe $n \in \mathbb{N}$ alle $k \in \mathbb{N}$ mit $0 \leq k \leq n$ als mögliche Ausgaben liefert.*

Lösung: Eine Lösung ist z.B.

$$
\begin{aligned}
A = \{1\colon\ & R\ \underline{\text{then}}\ 2,\\
2\colon\ & \underline{\text{if}}\ t\ \underline{\text{then}}\ 3\ \underline{\text{else}}\ 1\\
3\colon\ & L\ \underline{\text{then}}\ 4,\\
4\colon\ & \underline{\text{if}}\ t\ \underline{\text{then}}\ 0\ \underline{\text{else}}\ 5,\\
5\colon\ & L\ \underline{\text{then}}\ 6,\\
6\colon\ & \underline{\text{if}}\ t\ \underline{\text{then}}\ 0\ \underline{\text{else}}\ 5,\\
4\colon\ & \underline{\text{if}}\ t\ \underline{\text{then}}\ 0\ \underline{\text{else}}\ 7,\\
7\colon\ & d_B\ \underline{\text{then}}\ 8,\\
8\colon\ & L\ \underline{\text{then}}\ 4 \qquad \}.
\end{aligned}
$$

Die Idee dabei ist, dass zuerst die Eingabe vollständig gelesen wird (1.-3.). Sodann wird bei 4. nichtdeterministisch entschieden, ob die Eingabe gleich der Ausgabe sein soll (Block 4., 5., 6.), oder ob die Eingabe verkleinert werden soll (Block 4., 7., 8.). In letzterem Fall kann diese nichtdeterministische Entscheidung wiederholt getroffen werden (siehe Sprung nach 4 in 8.).

Übung 11.3 *Betrachten Sie das folgende Problem k-**FARB** (k-Graphfärbbarkeitsproblem):*

[1] Der Nichtdeterminismus wird hier modelliert durch die mögliche Nichteindeutigkeit der Kennmarken von Anweisungen.

- *Gegeben: Ein ungerichteter Graph $G = (P, K)$ mit Punktemenge $P = \{1, \ldots, m\}$ und Kantenmenge K, und eine natürliche Zahl $k \geq 1$. G heißt **k-färbbar**, wenn es (Farben) $a_1, a_2, \ldots, a_m \in \{1, \ldots, k\}$ gibt, so daß für alle Kanten $(i, j) \in K$ (mit $i \neq j$) stets $a_i \neq a_j$ gilt (d.h., je zwei mit einer Kante verbundene Punkte müssen verschiedene Farben haben).*

- *Frage: Ist G k-färbbar?*

Geben Sie einen möglichst effizienten Algorithmus an (auf informelle, aber doch präzise Art und Weise), der das k-Graphfärbbarkeitsproblem für den Spezialfall $k = 2$ entscheidet (d.h. einen Entscheidungsalgorithmus für 2-FARB). Welche Zeitkomplexität hat Ihr Algorithmus für 2-FARB?

Lösung: Die Grundidee zu einer effizienten Lösung von 2-FARB besteht in der Beobachtung, daß wenn ein bestimmer Punkt bereits gefärbt ist, alle mit diesem Punkt durch eine Kante verbundenen Punkte automatisch mit der Komplementärfarbe eingefärbt werden müssen (dieser Determinismus ist nicht mehr gegeben, wenn man $k \geq 3$ Farben zur Verfügung hat!).
Ein einfacher effizienter Algorithmus zur Lösung von 2-FARB ist etwa folgender (wobei wir die beiden Farben der Einfachheit wegen mit *rot* und *grün* bezeichnen:

(1) Beginne bei einem beliebigen Punkt p_1 (z.B. 1) und färbe diesen *rot* (genauso könnten wir mit *grün* beginnen).

(2) Betrachte alle mit p_1 durch eine Kante in K verbundenen Punkte p_i, $1 \leq i \leq m$, und färbe diese *grün*. Falls es eine Kante von p_1 nach p_1 gibt, ist die Antwort NEIN.

(3) Wiederhole Schritt (2) analog, statt mit p_1 für alle in (2) *grün* gefärbten Punkte und dieses Mal mit Farbe *rot*. Falls bei dieser Färbung eine Inkonsistenz auftritt (d.h. ein bereits *grün*) gefärbter Punkt *rot* wird), ist die Antwort NEIN.

(4) Wiederhole sinngemäß (3), d.h. Einfärbung aller Punkte, die mit einem zuletzt eingefärbten Punkt durch eine Kante verbundenen sind, durch die entspechende Komplementärfarbe, und zwar solange wie möglich.

(5) Falls dabei eine Färbungsinkonsistenz auftritt, ist die Antwort NEIN. Falls nicht, sind alle betrachteten Punkte des Graphen konsistent gefärbt. Jetzt gibt es 2 Fälle:

    (a) Alle Punkte des Graphen sind (konsistent) gefärbt. Dann ist die Antwort JA.

    (b) Falls nicht, haben wir erst einen unabhängigen Teilgraphen von G (d.h. einen solchen, von dem es keine Kanten zu anderen Punkten von G gibt) bearbeitet. In diesem Fall müssen wir das gesamte Verfahren für alle weiteren unabhängigen Teilgraphen von G entsprechend wiederholen (wobei die *Startfarbe* jeweils beliebig ist). Falls die Färbung aller unabhängigen Teilgraphen von G konsistent möglich ist, so ist die Antwort JA, ansonsten NEIN.

Die Korrektheit des Verfahrens ist offensichtlich, wenn man berücksichtigt, daß die Problemstellung symmetrisch in den beiden Farben ist.
Für die Komplexitätsanalyse nehmen wir als sinnvolles Maß n für die Größe der Eingabe die Größe der *Adjazenzmatrix* $A = (k_{ij})$ (mit $k_{ij} = \begin{cases} 1, & (i, j) \in K \\ 0, & \text{sonst} \end{cases}$) von G. Unter der (ebenfalls sinnvollen) Annahme, daß der Zugriff auf Matrixelemente und ihr Vergleich in konstanter Zeit erfolgt, und mit der Beobachtung, daß wir maximal alle Matrixelemente inspizieren müssen, ergibt sich damit eine Zeitkomplexität von $O(n)$ für den Algorithmus. Damit hat auch das Entscheidungsproblem 2-FARB Zeitkomplexität $O(n)$, da wir mindestens die Eingabe lesen müssen.

Übung 11.4 *Weisen Sie nach, dass das k-Graphfärbbarkeitsproblem k-FARB aus Übungsaufgabe 11.3 in* NP *liegt, indem Sie einen nichtdeterministischen polynomiellen Entscheidungsalgorithmus dafür angeben (skizieren).*

Lösung: Das nichtdeterministische Verfahren arbeitet wie folgt: Rate zu Beginn eine konkrete Färbung des Graphen bzw. genauer seiner Knoten. Überprüfe sodann deterministisch, ob diese Färbung im beschriebenen Sinne konsistent ist. Die Überprüfung letzterer Eigenschaft ist klarerweise polynomiell möglich (sogar in linearer Zeit).

Übung 11.5 *Weisen Sie nach, dass die Aussage*

> *Wenn es eine polynomiell entscheidbare Sprache L und eine NP-vollständige Sprache L' gibt mit $L' \leq_p L$, dann gilt* $\mathrm{P} = \mathrm{NP}$.

richtig ist.

Lösung: Nach Definition von NP-Vollständigkeit gilt: L' ist NP-vollständig gdw. $L' \in \mathrm{NP}$ und $L'' \leq_p L'$ für alle $L'' \in \mathrm{NP}$. Wenn nun $L \in \mathrm{P}$ und L' NP-vollständig mit $L' \leq_p L$, so heißt letzteres, dass es eine polynomielle Reduktion von $L' \in \mathrm{NP}$ auf $L \in \mathrm{P}$ gibt, also liegt insbesondere L' auch in P (siehe auch Satz 5.3.2). Aus der NP-Vollständigkeit von L' folgt dann: $L'' \leq_p L' \leq_p L$ für alle $L'' \in \mathrm{NP}$, also auch $L'' \in \mathrm{P}$ für alle $L'' \in \mathrm{NP}$, und daher $\mathrm{P} = \mathrm{NP}$.