SS 2015

Zentralübung zur Vorlesung Theoretische Informatik

Dr. Werner Meixner

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2015SS/theo/uebung/

9. Juli 2015

ZÜ X

Übersicht:

1. Übungsbetrieb Fragen, Probleme?
2. Thema Komplexitätsklassen
3. Vorbereitung Blatt 12

1. Fragen, Probleme?

Aktuelle Fragen?

2. Thema Komplexitätsklassen

Definitionen aus der Vorlesung

M deterministische, mehrbändige Turingmaschine:

$$\text{(D)TIME}_M(w) = \text{Anzahl der Schritte von } M \text{ bei Eingabe } w \text{ (ggf } \infty)$$

$$\text{(D)TIME}_M(n) = \max\{\text{DTIME}_M(w);\ |w| = n\};\ n \in \mathbb{N}_0$$

$$\text{(D)SPACE}_M(w) = \text{max. Anzahl der Arbeitsbandfelder, die } M \text{ bei Eingabe } w \text{ pro Arbeitsband besucht (ggf } \infty)$$

$$\text{(D)SPACE}_M(n) = \max\{\text{DSPACE}_M(w);\ |w| = n\};\ n \in \mathbb{N}_0$$

M nichtdeterministische, mehrbändige Turingmaschine:

$\mathsf{NTIME}_M(w) =$ Anzahl der Schritte einer kürzesten akzeptierenden Berechnung von M bei Eingabe w (ggf ∞)

$$\mathsf{NTIME}_M(n) = \max\{\mathsf{NTIME}_M(w);\ |w| = n\};\ n \in \mathbb{N}_0$$

$\mathsf{NSPACE}_M(w) =$ Anzahl der Arbeitsbandfelder, die eine akzeptierende Berechung von M bei Eingabe w pro Arbeitsband mindestens besucht (ggf ∞)

$$\mathsf{NSPACE}_M(n) = \max\{\mathsf{NSPACE}_M(w);\ |w| = n\};\ n \in \mathbb{N}_0$$

Wichtige Komplexitätsklassen

Sei $T(n)$ eine Zeitschranke, $S(n)$ eine Platzschranke. Wir definieren die folgenden Komplexitätsklassen:

1. DTIME($T(n)$) ist die Klasse aller Probleme, die von einer deterministischen Turingmaschine in Zeit $T(n)$ erkannt werden können.
2. NTIME($T(n)$) ist die Klasse aller Probleme, die von einer nichtdeterministischen Turingmaschine in Zeit $T(n)$ akzeptiert werden können.
3. DSPACE($S(n)$) ist die Klasse aller Probleme, die von einer deterministischen Turingmaschine in Platz $S(n)$ erkannt werden können.
4. NSPACE($S(n)$) ist die Klasse aller Probleme, die von einer nichtdeterministischen Turingmaschine in Platz $S(n)$ akzeptiert werden können.

Davon abgeleitet sind die folgenden Komplexitätsklassen:

1. $$\mathcal{P} = \bigcup_{c>0,k>0} \mathsf{DTIME}(cn^k);$$

2. $$\mathcal{NP} = \bigcup_{c>0,k>0} \mathsf{NTIME}(cn^k);$$

3. $$\mathcal{L} = \bigcup_{c>0} \mathsf{DSPACE}(c \log n);$$

4. $$\mathcal{NL} = \bigcup_{c>0} \mathsf{NSPACE}(c \log n);$$

Davon abgeleitet sind die folgenden Komplexitätsklassen:

5. $$\mathsf{PSPACE} = \bigcup_{c>0,k>0} \mathsf{DSPACE}(cn^k);$$

6. $$\mathsf{NPSPACE} = \bigcup_{c>0,k>0} \mathsf{NSPACE}(cn^k);$$

3. Vorbereitung Blatt 12

3.1 VA 1

Warum kann man den Satz von Rice auf die folgende Menge nicht anwenden?

$$L = \{w \in \Sigma^* \,;\, \forall n \in \mathbb{N}_0 : \, \varphi_w(n) = \bot \text{ und } w \text{ ist ein Palindrom}\} \,.$$

Lösung

Hier ist der Satz von Rice nicht anwendbar, denn dafür müsste es eine Funktionenmenge F geben, so dass $L = \{w\,;\,\varphi_w \in F\}$.

Es ist aber nicht ausgeschlossen, dass es Wörter v und w gibt, so dass $\varphi_v = \varphi_w$ und v ein Palindrom ist, aber w keines.

Ob es solche Wörter tatsächlich gibt, hängt von der konkreten Codierung von Turingmaschinen in Wörter ab.

3.2 VA 2

1. Wir betrachten das Postsche Korrespondenzproblem
   $P = (\,(1, c1), (abc, ab)\,)$.
   Bestimmen Sie *alle* Lösungen von P !
2. Sei $P = (p_1, p_2)$ ein Postsches Korrespondenzproblem über einem beliebigen Alphabet Σ mit $p_i = (x_i, y_i)$ und
   $|\,|x_i| - |y_i|\,| = 1$ für $i = 1, 2$.
   Zeigen Sie, dass P entscheidbar ist!

1. Wir betrachten das Postsche Korrespondenzproblem
   $P = (\,(1, c1), (abc, ab)\,)$.
   Bestimmen Sie *alle* Lösungen von P !

Lösung

Für die Menge L der Lösungen gilt
$L = \{(i_1 i_2)^n\,;\, n \in \mathbb{N}_0, n \neq 0, i_1 = 2, i_2 = 1\}$.

2. Sei $P = (p_1, p_2)$ ein Postsches Korrespondenzproblem über einem beliebigen Alphabet Σ mit $p_i = (x_i, y_i)$ und $\big||x_i| - |y_i|\big| = 1$ für $i = 1, 2$.
   Zeigen Sie, dass P entscheidbar ist!

Lösung

Es gibt genau dann eine Lösung von P, wenn $1, 2$ oder $2, 1$ eine Lösung ist, d. h. wenn entweder $x_1x_2 = y_1y_2$ oder $x_2x_1 = y_2y_1$ gilt.

Beweis

Wir nehmen an, dass $i_1i_2 \dots i_k$ eine Lösung von P ist.

Zunächst gilt sicher $k \geq 2$, denn aus der Längenbedingung für die p_i folgt, dass weder p_1 noch p_2 selbst schon eine Lösung ist.

Falls $i_1 \neq i_2$, dann folgt $|x_{i_1} x_{i_2}| = |y_{i_1} y_{i_2}|$.

Mithin ist bereits $i_1 i_2$ eine Lösung, d. h. $1, 2$ oder $2, 1$ ist Lösung.

Sei nun $i_1 = i_2$ und o. B. d. A. $i_1 = i_2 = 1$.

Wir nehmen ebenfalls o. B. d. A. an, dass y_1 ein Präfix von x_1 ist, d. h. $x_1 = u_1 u_2 \ldots u_n u_{n+1}$ und $y_1 = u_1 u_2 \ldots u_n$ mit $u_1, \ldots, u_{n+1} \in \Sigma$.

Da $i_1 i_2$ Teil einer Lösung ist, gilt

$$\begin{aligned} x_1 x_1 &= y_1 y_1 u_n u_{n+1} \quad \text{und} \\ u_1 u_2 \ldots u_n u_{n+1} u_1 u_2 \ldots u_n u_{n+1} &= u_1 u_2 \ldots u_n u_1 u_2 \ldots u_n u_n u_{n+1} \end{aligned}$$

Durch buchstabenweisen Vergleich folgt

$$u_{n+1} = u_1 = u_2 = \ldots = u_{n-1} = u_n \,,$$

Wir erhalten $p_1 = (a^{n+1}, a^n)$ für irgendeinen Buchstaben $a \in \Sigma$.

In der Sequenz $i_1 i_2 \ldots i_k$ muss es eine Teilsequenz $i_1 i_2 \ldots i_l$ geben, die selbst Lösung ist und für die gilt $i_{l-1} = 2, i_l = 2$. Dann zeigt man analog $p_2 = (b^m, b^{m+1})$.

Da nun p_1 und p_2 irgendwo gematcht werden müssen, folgt $a = b$. Mithin folgt, dass $1, 2$ eine Lösung ist. Es folgt sogar, dass $2, 1$ ebenfalls eine Lösung ist.

3.3 VA 3

Zeigen Sie, dass die polynomielle Reduzierbarkeit $\leq_p$ eine transitive Relation ist. Polynomielle Reduzierbarkeit bedeutet, dass die Reduktionsfunktion in polynomieller Zeit berechenbar ist.

Lösung

Seien $\Sigma_1, \Sigma_2, \Sigma_3$ Alphabete und $A \subseteq \Sigma_1^*, B \subseteq \Sigma_2^*, C \subseteq \Sigma_3^*$, so dass $A \leq_p B$ und $B \leq_p C$ gilt.

Wir zeigen, dass dann auch $A \leq_p C$ gilt, wie folgt.

Nach Definition der Reduzierbarkeit gibt es berechenbare Funktionen $f : \Sigma_1^* \to \Sigma_2^*$ und $g : \Sigma_2^* \to \Sigma_3^*$ (beide total in Σ_1^* bzw. Σ_2^*) mit entsprechenden, f bzw. g berechnenden DTM's F bzw. G, sowie Polynome p, q, so dass gilt

$$\begin{aligned} \forall x \in \Sigma_1^*. \quad \mathsf{TIME}_F(x) &\leq p(|x|)\,, \\ \forall x \in \Sigma_2^*. \quad \mathsf{TIME}_G(x) &\leq q(|x|)\,. \end{aligned}$$

und außerdem noch $f^{-1}(B) = A$ und $g^{-1}(C) = B$ gilt.

Wir zeigen nun, dass die Funktionskomposition $g \circ f$ die Menge A polynomiell berechenbar auf C reduziert.

Zunächst gilt $(g \circ f)^{-1}(C) = f^{-1}(g^{-1}(C)) = f^{-1}(B) = A$, d. h. $x \in A \iff (g \circ f)(x) \in C$.

Dann gibt es eine Turingmaschine $K(F, G)$ die die Komposition von F und G im Wesentlichen als Simulation von F angewandt auf x und nachfolgend G angewandt auf die Ausgabe $f(x)$ ausführt.

Man kann die Schrittzahl der Simulation von F angewandt auf x mit $p(|x|)$ abschätzen.

Die Schrittzahl der Simulation von G angewandt auf $f(x)$ wird mit $q(|f(x)|)$ abgeschätzt. Da die Länge von $f(x)$ sicher höchstens gleich $|x| + p(|x|)$ ist, erhalten wir für die Abschätzung der Simulation von G angewandt auf $f(x)$ die Schranke $q(|x| + p(|x|))$.

Allerdings wird die Maschine K für die Hintereinanderschaltung von F und G noch Berechnungsschritte ausführen, die aber sicher so angelegt werden können, dass sie nur linear von $|x|$ und $|f(x)|$ abhängen. Eine zusätzliche Abhängikeit von $|(g \circ f)(x)|$ kann man vermeiden.

Zusammenfassend erhalten wir

$$\forall x \in \Sigma_1^*. \quad \mathsf{TIME}_K(x) \quad \leq \quad p(|x|) + q(|x| + p(|x|)) + c(|x| + p(|x|))\,.$$

Offenbar ist $g \circ f$ polynomiell beschränkt.

3.4 VA 4

Beantworten Sie kurz die folgenden Fragen:

1. Ist TIME_M für jede deterministische Turingmaschine M berechenbar?
2. PSPACE ist die Klasse all jener Probleme, die eine DTM mit ,,polynomiell viel Band in Abhängigkeit der Länge der Eingabe“ lösen kann. Gilt $\mathcal{P} \subseteq$ PSPACE?

Lösung

1. Nein, denn dann wäre das Halteproblem entscheidbar.
2. Ja, denn in polynomieller Zeit kann nur polynomiell viel Band beschrieben werden. Ob PSPACE $\subseteq \mathcal{P}$ gilt, ist ein offenes Problem.

3.5 VA 5

Beweisen Sie:

1. $\mathcal{P}$ ist abgeschlossen unter Komplement.
2. Das Problem, zu entscheiden, ob ein gegebener Graph ein Dreieck enthält, ist in $\mathcal{P}$.

Lösung

1. Sei $A \in \Sigma^*$ in $\mathcal{P}$ und sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ eine Turingmaschine, die A in polynomieller Zeit entscheidet, d.h. die charakteristische Funktion χ_A berechnet. Dann ist $M' = (Q \cup \{\overline{q}\}, \Sigma, \Gamma, \delta', q_0, \Box, \{\overline{q}\})$ mit

$$\delta'(q,a) = \begin{cases} (\overline{q}, 0, N) & \text{falls } q \in F \text{ und } a = 1 \\ (\overline{q}, 1, N) & \text{falls } q \in F \text{ und } a = 0 \\ \delta(q,a) & \text{sonst} \end{cases}$$

eine Turingmaschine, die $\overline{A}$ entscheidet, und zwar ebenso in polynomieller Zeit. Daher ist auch $\overline{A}$ in $\mathcal{P}$.

2. Folgender Algorithmus entscheidet dieses Problem:

   Prüfe für je drei verschiedene Knoten des Graphen, ob sie untereinander verbunden sind.

   Da es höchstens n^3 verschiedene Tripel von Knoten eines Graphen mit insgesamt n Knoten gibt und der Test auf Verbundenheit dreier Knoten in polynomieller Zeit möglich ist, ist das Problem des Dreiecks in einem Graphen in $\mathcal{P}$.

Viel Erfolg in der Endterm!!